2017

Improving Facial Action Unit Recognition Using Convolutional Neural Networks

Shizhong Han
University of South Carolina

Improving Facial Action Unit Recognition Using Convolutional Neural Networks

by

Shizhong Han

Bachelor of Information and Computation Science
Hunan University China 2008

Master of Computer Science
Wuhan University China 2010

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosphy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2017

Accepted by:

Yan Tong, Major Professor

Srihari Nelakuditi, Committee Member

Michael Huhns, Committee Member

Song Wang, Committee Member

XiaoFeng Wang, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Acknowledgments

First I would like to gratefully thanks my adviser, Prof. Yan Tong, for her guidance, patience, and encouragement during my graduate studies. She gave me great encouragement many times when I was discouraged by the experiment results. She gave me confidence when I want to give up the research project. She also gave me great freedom and a lot of helpful suggestions for the research work.

I want to thank my dissertation committee members, Prof. Michael Huhns, Prof. Srihari Nelakuditi, Prof. Song Wang, and Prof. Xiaofeng Wang, for their invaluable advice on my work. It is a honor to have them serve in my committee.

I want to thank my colleagues including: Ping Liu, Zibo Meng, James O'Reilly, Shehab Khan, Jie Cai, Zhiyuan Li, Dazhou Guo, Kang Zheng, Hongkai Yu, Yang Mi, Haozhou Yu, Hao Guo and other members in our research group. It is my pleasure to meet them and have a chance to cooperate with them.

Finally, my special thanks go to my family. I would like to express my earnest gratitude to my wife Haixia Li. I owe a lot of gratitude to her who accompanied and supported me throughout these years. I also thank my parents for their love and sacrifices when I am pursuing my goal. I cannot thank my parents-in-law enough for giving me the freedom to pursue my interests.

This research was supported by the National Science Foundation under CAREER Award IIS-1149787.

Abstract

Recognizing facial action units (AUs) from spontaneous facial expression is a challenging problem, because of subtle facial appearance changes, free head movements, occlusions, and limited AU-coded training data. Most recently, convolutional neural networks (CNNs) have shown promise on facial AU recognition. However, CNNs are often overfitted and do not generalize well to unseen subject due to limited AU-coded training images. In order to improve the performance of facial AU recognition, we developed two novel CNN frameworks, by substituting the traditional decision layer and convolutional layer with the incremental boosting layer and adaptive convolutional layer respectively, to recognize the AUs from static image.

First, in order to handle the limited AU-coded training data and reduce the overfitting, we proposed a novel Incremental Boosting CNN (IB-CNN) to integrate boosting into the CNN via an incremental boosting layer that selects discriminative neurons from the lower layer and is incrementally updated on successive mini-batches. In addition, a novel loss function that accounts for errors from both the incremental boosted classifier and individual weak classifiers was proposed to fine-tune the IB-CNN. Experimental results on four benchmark AU databases have demonstrated that the IB-CNN yields significant improvement over the traditional CNN and the boosting CNN without incremental learning, as well as outperforming the state-of-the-art CNN-based methods in AU recognition. The improvement is more impressive for the AUs that have the lowest frequencies in the databases.

Second, all current CNNs use predefined and fixed convolutional filter size. However, AUs activated by different facial muscles cause facial appearance changes at

different scales and thus favor different filter sizes. The traditional strategy is to experimentally select the best filter size for each AU in each convolutional layer, but it suffers from expensive training cost, especially when the networks become deeper and deeper. We proposed a novel Optimized Filter Size CNN (OFS-CNN), where the filter sizes and weights of all convolutional layers are learned simultaneously from the training data along with learning convolutional filters. Specifically, the filter size is defined as a continuous variable, which is optimized by minimizing the training loss. Experimental results on four AU-coded databases and one spontaneous facial expression database outperforms traditional CNNs with fixed filter sizes and achieves state-of-the-art recognition performance. Furthermore, the OFS-CNN also beats traditional CNNs using the best filter size obtained by exhaustive search and is capable of estimating optimal filter size for varying image resolution.

Table of Contents

List of Figures

Chapter 1

Introduction

Facial behavior is the most powerful means to express and perceive the emotions and intentions of a human. The Facial Action Coding System (FACS), developed by Ekman and Friesen [11], is a comprehensive and objective system for measuring facial behavior. With FACS, facial behavior can be described by combinations of facial action units (AUs), each of which is anatomically related to the contraction of a set of facial muscles. In addition to applications in human behavior analysis, an automatic AU recognition system has great potential to advance emerging applications in human-computer interaction (HCI), such as online/remote education, interactive games, and intelligent transportation, as well as to push the frontier of research in psychology. However, recognizing facial AUs from spontaneous facial expressions is challenging because of the following:

- First, facial actions are rich, complex, and most involve subtle appearance changes. Thousands of distinct nonrigid facial muscular movements (different AU combinations) have been observed so far [56], and most of them differ subtly in a few facial features.

- Second, AU-coded training data is limited in terms of subjects and images/videos compared with many other tasks such as object detection and recognition, because of AU labeling performed by certified AU labeler requires expert knowledge and is highly labor-intensive with limited throughput.

- Third, the training data is highly unbalanced with a small percentage of positive data for some AUs, which increases the difficulty to train a model with good generalization capability.

In order to handle those challenges and improve the performance of facial AU recognition, we developed two CNN frameworks by substituting the traditional decision layer and convolutional layer with the incremental boosting (IB) layer and the

optimal filter size (OPS) convolutional layer respectively. The whole framework can be optimized using stochastic gradient descent.

1.1 Related Work on Facial Action Unit Recognition

In this section, we will discuss the related work on facial AU recognition, especially the CNN based methods.

Extensive research has been conducted on recognizing AUs and their combinations from video sequences or static images by extracting the representative facial geometrical or appearance features as detailed in the surveys [50, 6, 76, 54]. Most of the existing work exploits a variety of hand-crafted features, which generally used magnitudes of a set of Gabor wavelets extracted at multiple scales and orientations either from the whole face region or at a few fiducial points [63, 5, 78, 77, 68, 65, 64], Haar wavelet features [68] considering the intensity difference of adjacent regions, and Scale Invariant Feature Transform (SIFT) features [72] extracted at a set of keypoints that are invariant to uniform scaling and orientation. Histograms of features extracted from a predefined facial grid have also been employed, such as histograms of Local Binary Patterns (LBPs) [58, 57, 67], Histograms of Oriented Gradients (HOG) [4], histograms of Local Phase Quantization (LPQ) features [25], and histograms of Local Gabor Binary Patterns (LGBP). In addition, spatiotemporal extensions of the aforementioned 2D features such as LBP-TOP [79], LGBP-TOP [2, 1], LPQ-TOP [25], and dynamic Haar-like features [69], which are usually calculated from three orthogonal planes, have been proposed to capture the spatiotemporal changes.

In addition to the human-crafted feature representations, features can also be learned in a data-driven manner by sparse coding [41, 49, 74, 70, 33, 39, 82] or deep learning [14, 42, 51, 53, 52, 34, 61, 35, 37, 26, 62, 17, 22]. As an over-complete representation learned from given input, sparse coding can capture a wide range of variations that are not targeted to a specific application and has achieved promising

results in facial expression recognition [74, 70, 33, 39, 82]. More recently, Nonnegative Sparse Coding (NNSC) [21] takes advantages of both sparse coding and Nonnegative Matrix Factorization (NMF) [30] and has been adopted in facial expression recognition [7, 81, 73, 38]. To become more adaptable to the real world that consists of a combination of edges [12], deep learning has been employed for facial expression recognition including deep belief network based approaches [51, 52, 34, 37] and convolutional neural network (CNN) based approaches [14, 42, 53, 61, 35, 26, 62, 17, 22]. Most of these deep-learning based methods took the whole face region as input and learned the high-level representations through a set of processing layers.

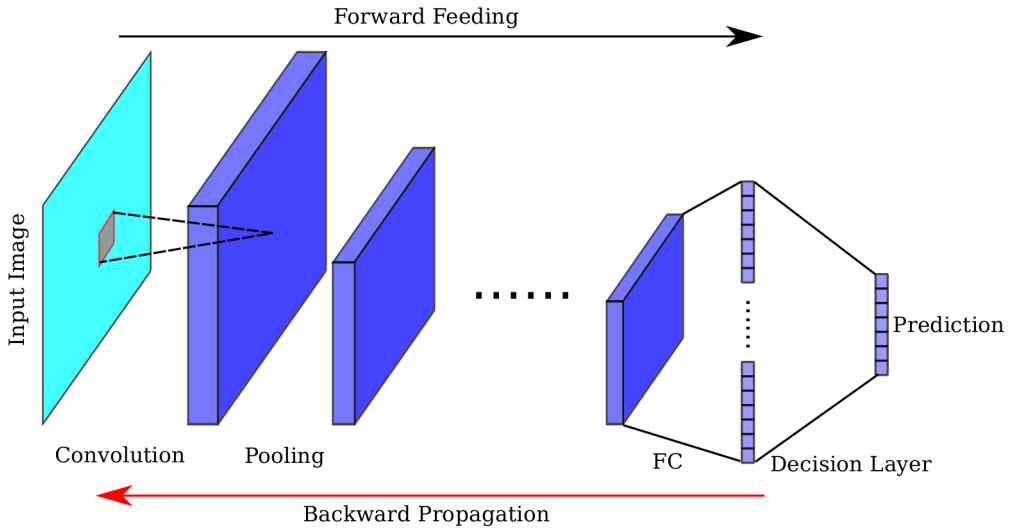

Figure 1.1 Overview of a standard convolutional neural network. FC is the fully connected layer. The input images are transformed to highly nonlinear representations through a set of processing layers.

1.1.1 CNN BASED METHODS

Among the feature learning based methods, CNNs [13, 53, 61, 35, 27, 17, 22] have attracted increasing attention. Gudi et al. [17] used a pre-processing method with local and global contrast normalization to improve the inputs of CNNs. Fasel [13] employed multi-size convolutional filters to learn multi-scale features. Liu et al [35] extracted

spatiotemporal features using the 3D CNN. Jung et al. [27] jointly fine-tuned temporal appearance and geometry features. Jaiswal and Valstar [22] integrated bi-directional long-short-term memory neural networks with the CNN to extract temporal features.

A CNN consists of a stack of one or more layers such as convolutional layers, pooling layers, rectification layers, fully connected layers, decision layers and loss layers such as in Figure 1.1. Those layers transform the input data to highly nonlinear representations. In the following section, we will focus on the work related to the design of decision layers and the selection of optimal convolutional filter sizes of CNNs.

Decision layer in CNN

Most CNN-based methods make decisions using the inner product of a fully connected layer. A few approaches developed new objective functions to improve recognition performance. Tang [47, 61] replaced the softmax loss function with a SVM for optimization. Hinton et al. [20] utilized a dropout technique to reduce overfitting by dropping out some neuron activations from the previous layer, which can be seen as an ensemble of networks sharing the same weights. However, the dropout process is random, regardless of the discriminative power of individual neurons. *In contrast, the proposed IB-CNN use the boosting (AdaBoost) algorithm to effectively select the more discriminative neurons and drop out noisy or redundant neurons, and at the same time to make the decision with the features from those selected neurons.*

Medera and Babinec [44] adopted incremental learning using multiple CNNs trained individually from different subsets, and additional CNNs are trained given new samples. Then, the prediction is calculated by weighted majority-voting of the outputs of all CNNs. However, each CNN may not have sufficient training data, which is especially true with limited AU-coded data. *Different from [44], the proposed IB-CNN has only one CNN trained along with an incremental strong classifier,*

where weak learners are updated over time by accumulating information from multiple batches to avoid overfitting. Liu et al. [37] proposed a boosted deep belief network for facial expression recognition, where each weak classifier is learned exclusively from an image patch. *In contrast, weak classifiers are selected from an fully connected layer in the proposed IB-CNN and thus, learned from the whole face.*

Convolution filter size in CNN

Most of the methods select the best filter sizes experimentally or by visualization. For facial expression recognition [29], a 5×5 filter size has better performance than other filter sizes. Demonstrated through visualization, 7×7 filters can capture more distinctive features than 11×11 filters on ImageNet dataset [75]. *However, with CNNs becoming deeper and deeper [60, 19], it's impractical to select the best filter sizes by experiments, because of the expensive training cost.* Concatenation of the activated features from different filter sizes can improve the performance. The inception model was proposed to concatenate the activation feature maps from filters of 1×1, 3×3, and 5×5 [60].

However, all of these aforementioned methods use predefined convolutional filter sizes. *The proposed OFS convolutional layer can optimize the filter size along with CNN training, which is especially desired when current networks become deeper and deeper. Since the size of filters in each layer can be optimized, the proposed method may achieve the same performance using a shallow CNN*

1.2 Scope of the Proposed Research

This research aims to improve facial analysis by developing: 1) an incremental boosting convolutional neural network to integrate boosting into the CNN via an incremental boosting layer, and 2) a method to optimize the convolutional filter size during CNN training.

In order to handle the limited AU-coded training data and improve the recognition rate, we first proposed a novel IB-CNN to integrate boosting into the CNN via an incremental boosting layer that selects discriminative neurons from the lower layers and is incrementally updated on successive mini-batches. In addition, a novel loss function that accounts for errors from both the incremental boosted classifier and individual weak classifiers was proposed to fine-tune the IB-CNN. Experimental results on four benchmark AU databases have demonstrated that the IB-CNN yields a significant improvement over the traditional CNN and the boosting CNN without incremental learning, as well as outperforming the state-of-the-art CNN-based methods in AU recognition. The improvement is more impressive for the AUs that have the lowest frequencies in the databases.

Second, all the current CNNs use predefined convolutional filter sizes, but different AUs cause appearance changes over different region sizes and therefore prefer different filter sizes. Some AUs can be recognized based on appearance changes in large regions such as the long nasolabial furrow caused by AU10 (upper lip raiser), while some AUs can be recognized in small regions such as short wrinkles in the skin above and below the lips and small bulges below the lower lip produced by AU23 (lip tightener). Motivated by those observation, we proposed a method to optimize the convolutional filter sizes. The filter size is a variable that can be learned from training data in contrast to the constant filter size in traditional CNNs, which can improve AU recognition performance as well as decrease the training cost incurred to select the best filter sizes experimentally.

The proposed methods are application independent, so it should generalize to other applications such as facial expression recognition, which will be done in future work.

1.3 Evaluation Databases

To evaluate the effectiveness of the proposed methods, extensive experiments have been conducted on the following benchmark AU-coded facial databases, i.e. CK database [28], FERA2015 SEMAINE database [66], FERA2015 BP4D database [66], DISFA dataset [43]. Because the limited subjects (less than 100) in the AU recognition subjects, the database of Static Facial Expressions in the Wild (SFEW) with more than 1000 subjects is used to show the effectiveness of proposed method.

The CK database [28] was evaluated to demonstrate generalization capability of the proposed method, since CK contains 486 image sequences from 97 subjects and has been widely used for evaluating the performance of AU recognition. 14 AUs are annotated frame-by-frame for training and evaluation. The 8-fold cross-validation stretagy is employed so that the subjects in different partitions are mutually exclusive.

The FERA2015 SEMAINE was used as a benchmark dataset for the FERA2015 AU recognition challenge, where videos were recorded in conversations between people and a virtual agent. The SEMAINE database contains spontaneous facial displays from 31 subjects, which were divided into two groups: 15 for training with 48,000 images and 16 for validation with 45,000 images. All the 44 AUs were manually labeled in the dataset, 6 AUs were selected in the challenge.

The FERA2015 BP4D database [66] was also a dataset for the FERA2015 challenge. The videos in the BP4D database consist of young adults responding to emotion elicitation tasks. There are 11 coded AUs from 41 subjects with a total of 146,847 images.

The DISFA database [43] contains stereo videos of 27 subjects covering different races, genders, and ages with a total of 130,814 images. Each subject was asked to watch a 4-minute video for eliciting emotion, and spontaneous facial activities with natural head movements were recorded. 12 AUs were manually labeled. The 9-fold cross-validation stretagy is employed so that the subjects in different partitions are

mutually exclusive.

The SFEW database [9] collects spontaneous facial expression in the wild with static images, which contain large head movements and have been widely used for facial expression recognition. The SFEW database includes 1,766 images, i.e. 958 for training, 436 for validation, and 372 for testing. Every image has one of seven expression labels including anger, disgust, fear, neutral, happy, sad, and surprise.

1.4 Structure of the Dissertation

This dissertation is arranged as follows. Chapter 2 presents a novel incremental boosting convolutional neural network that integrates the boosting algorithm into the CNN as an unified framework for optimization. Chapter 3 describes a method for optimizing the convolutional filter size in CNN. The contributions are summarized in Chapter 4.

Chapter 2

Incremental Boosting CNN

2.1 Motivation

In this work, an incremental boosting CNN was proposed to integrate the boosting algorithm into a CNN as an unified frame work for the facial action unit recognition.

Boosting, e.g., AdaBoost, is a popular ensemble learning technique, which combines many "weak" classifiers and has been demonstrated to yield better generalization performance in AU recognition [5]. Boosting can be integrated into the CNN such that discriminative neurons are selected and activated in each iteration of CNN learning. However, the boosting CNN (B-CNN) can overfit due to the limited training data in each mini-batch. Furthermore, the information captured in previous iteration/batch cannot be propagated, i.e., a new set of weak classifiers is selected in every iteration and the weak classifiers learned previously are discarded.

Inspired by incremental learning, we proposed a novel *Incremental Boosting CNN (IB-CNN)*, which aims to accumulate information in B-CNN learning when new training samples appear. As shown in Fig. 2.1, a batch of images is employed in each iteration of CNN learning. The outputs of the fully-connected (FC) layer are employed as features; a subset of features (the blue nodes), which is discriminative for recognizing the target AU in the current batch, is selected by boosting. Then, these selected features are combined with the ones selected previously (the red nodes) to form an incremental strong classifier. The weights of active features, i.e., both the blue and the red nodes, are updated such that the features selected most of the time have higher weights. Finally, a loss, i.e., the overall classification error from both weak classifiers and the incremental strong classifier, is calculated and backpropagated to fine-tune the CNN iteratively. The proposed IB-CNN has a complex decision boundary due to boosting and is capable of alleviating the overfitting problem for the mini-batches by taking advantage of incremental learning.

In summary, this work has three major contributions. *(1) Feature selection and classification are integrated with CNN optimization in a boosting CNN framework. (2)*

A novel incremental boosted classifier is updated iteratively by accumulating information from multiple batches. (3) A novel loss function, which considers the overall classification error of the incremental strong classifier and individual classification errors of weak learners, is developed to fine-tune the IB-CNN.

Experimental results on four benchmark AU-coded databases, i.e., Cohn-Kanade (CK) [28], FERA2015 SEMAINE [66], FERA2015 BP4D [66], and Denver Intensity of Spontaneous Facial Action (DISFA) [43] databases have demonstrated that the proposed IB-CNN significantly outperforms the traditional CNN model as well as the state-of-the-art CNN-based methods for AU recognition. Furthermore, the performance improvement of the infrequent AUs is more impressive, which demonstrates that the proposed *IB-CNN* is capable of improving CNN learning with limited training data. In addition, the performance of IB-CNN is not sensitive to the number of neurons in the FC layer and the learning rate, which are favored traits in CNN learning.

2.2 Methodology

As illustrated in Figure 2.1, an IB-CNN model is proposed to integrate boosting with the CNN at the decision layer with an incremental boosting algorithm, which selects and updates weak learners over time as well as constructs an incremental strong classifier in an online learning manner. There are three major steps for incremental boosting: selecting and activating neurons (blue nodes) from the FC layer by boosting, combining the activated neurons from different batches (blue and red nodes) to form an incremental strong classifier, and fine-tuning the IB-CNN by minimizing the proposed loss function. In the following, we start with a brief review of CNNs and then, describe the three steps of incremental boosting in detail.

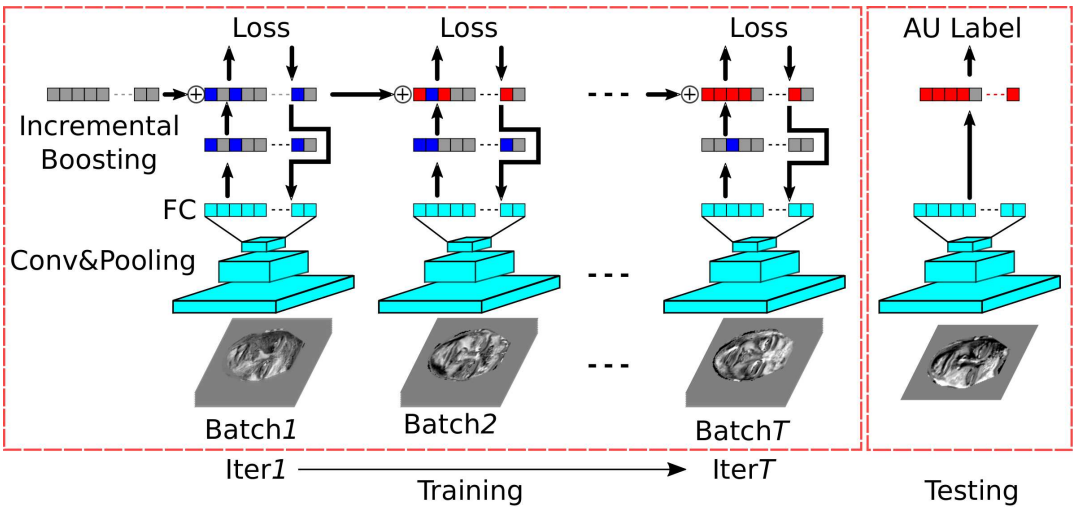

Figure 2.1 An overview of Incremental Boosting CNN. An incremental boosted classifier is trained iteratively. Outputs of the FC layer are employed as input features and a subset of features (the blue nodes) are selected by boosting. The selected features in the current iteration are combined with those selected previously (the red nodes) to form an incremental strong classifier. A loss is calculated based on the incremental classifier and propagated backward to fine-tune the CNN parameters. The gray nodes are inactive and thus, not selected by the incremental strong classifier. Given a testing image, features are calculated via the CNN and fed to the boosted classifier to predict the AU label. Best viewed in color.

2.2.1 A Brief Review of CNNs

A CNN consists of a stack of layers such as convolutional layers, pooling layers, rectification layers, FC layers, and a decision layer and transforms the input data into a highly nonlinear representation. Ideally, learned filters should activate the image patches related to the recognition task, i.e., detecting AUs in this work. Neurons in an FC layer have full connections with all activations in the previous layer. Finally, high-level reasoning is done at the decision layer, where the number of outputs is equal to the number of target classes. The score function used by the decision layer is generally the inner product of the activations in the FC layer and the corresponding weights. During CNN training, a loss layer is employed after the decision layer to specify how to penalize the deviations between the predicted and true labels, where

different types of loss functions have been employed, such as softmax, SVM, and sigmoid cross entropy. In this work, we substitute the inner-product score function with a boosting score function to achieve a complex decision boundary.

2.2.2 Boosting CNN

In a CNN, a mini-batch strategy is often used to handle large training data. Let $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_M]$ be the activation features of a batch with M training images, where the dimension of the activation feature vector $\mathbf{x}_i$ is K, and $\mathbf{y} = [y_1, ..., y_M]$, $y_i \in \{-1, 1\}$ is a vector storing the ground truth labels. With the boosting algorithm, the prediction is calculated by a strong classifier $H(\cdot)$ that is the weighted summation of weak classifiers $h(\cdot)$ as follows:

$$H(\mathbf{x}_i) = \sum_{j=1}^{K} \alpha_j h(x_{ij}, \lambda_j);\ h(x_{ij}, \lambda_j) = \frac{f(x_{ij}, \lambda_j)}{\sqrt{f(x_{ij}, \lambda_j)^2 + \eta^2}} \tag{2.1}$$

where $x_{ij} \in \mathbf{x}_i$ is the j^{th} activation feature of the i^{th} image. Each feature corresponds to a candidate weak classifier $h(x_{ij}, \lambda_j)$ with output in the range of (-1,1). $\frac{f(\cdot)}{\sqrt{f(\cdot)^2+\eta^2}}$ is used to simulate a $sign(\cdot)$ function to compute the derivative for gradient descent optimization. In this work, $f(x_{ij}, \lambda_j) \in \mathbb{R}$ is defined as a one-level decision tree (a decision stump) with the threshold of λ_j, which has been widely used in AdaBoost. The parameter η in Eq. 2.1 is employed to control the slope of function $\frac{f(\cdot)}{\sqrt{f(\cdot)^2+\eta^2}}$ and can be set according to the distribution of $f(\cdot)$ as $\eta = \frac{\sigma}{c}$, where σ is the standard deviation of $f(\cdot)$ and c is a constant. In this work, η is empirically set to $\frac{\sigma}{2}$. $\alpha_j \geq 0$ is the weight of the j^{th} weak classifier and $\sum_{j=1}^{K} \alpha_j = 1$. When $\alpha_j = 0$, the corresponding neuron is inactive and will not go through the feedforward and backpropagation process.

Traditional boosting algorithms only consider the loss of the strong classifier, which can be dominated by some weak classifiers with large weights, potentially leading to overfitting. To account for classification errors from both the strong classifier

and the individual classifiers, the loss function is defined as the summation of a strong-classifier loss and a weak-classifier loss as follows:

$$\varepsilon^B = \beta\varepsilon^B_{strong} + (1-\beta)\varepsilon_{weak} \tag{2.2}$$

where $\beta \in [0,1]$ balances the strong-classifier loss and the weak-classifier loss.

The strong-classifier loss is defined as the Euclidean distance between the prediction and the groundtruth label:

$$\varepsilon^B_{strong} = \frac{1}{M}\sum_{i=1}^{M}(H(\mathbf{x}_i) - y_i)^2 \tag{2.3}$$

The weak-classifier loss is defined as the summation of the individual losses of all weak classifiers:

$$\varepsilon_{weak} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{\substack{1\le j\le K\\ \alpha_j>0}}\Big[h(x_{ij},\lambda_j) - y_i\Big]^2 \tag{2.4}$$

where the constraint $\alpha_j > 0$ excludes inactive neurons when calculating the loss.

Driven by the loss ε^B defined in Eq. 2.2, the B-CNN can be iteratively fine-tuned by backpropagation as illustrated in the top of Figure 2.2. However, the information captured previously, e.g., the weights and thresholds of the active neurons, is discarded for a new batch. Due to limited data in each mini-batch, the trained B-CNN can be overfitted.

2.2.3 Incremental Boosting

Incremental learning can help to improve the prediction performance and to reduce overfitting. As illustrated in the bottom of Figure 2.2, both of the blue nodes selected in the current iteration and the red nodes selected previously are incrementally combined to form an incremental strong classifier H^t_I at the t^{th} iteration:

$$H^t_I(\mathbf{x}_i) = \frac{(t-1)H^{t-1}_I(\mathbf{x}_i) + H^t(\mathbf{x}_i)}{t} \tag{2.5}$$

where $H^{t-1}_I(\mathbf{x}_i)$ is the incremental strong classifier obtained at the $(t-1)^{th}$ iteration; and $H^t(\mathbf{x}_i)$ is the boosted strong classifier estimated in the current iteration.

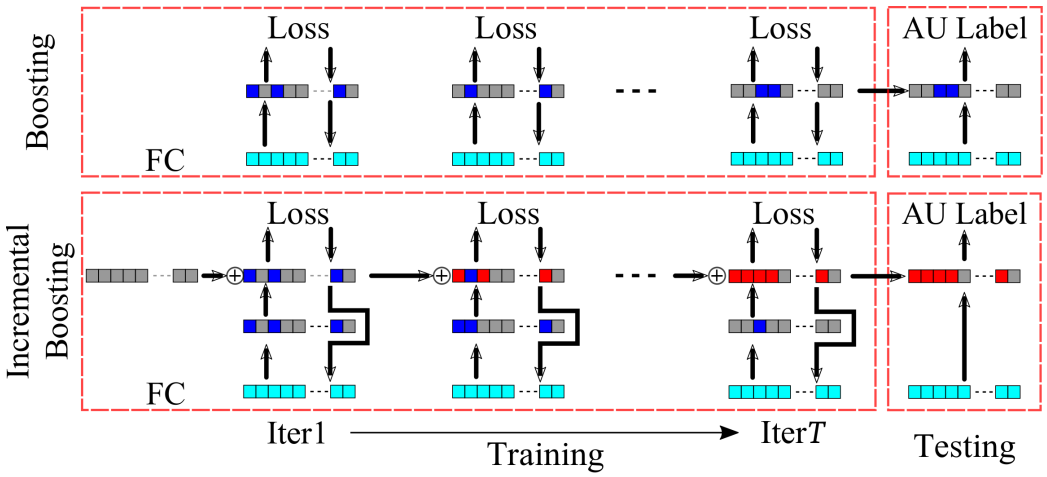

Figure 2.2 A comparison of the IB-CNN and the B-CNN structures. For clarity, the illustration of IB-CNN or B-CNN starts from the FC layer (the cyan nodes). The blue nodes are active nodes selected in the current iteration; the red nodes are the active nodes selected from previous iterations; and the gray nodes are inactive.

Algorithm 1 Incremental Boosting Algorithm for the IB-CNN

Input: The number of iterations (mini-batches) T and activation features $\mathbf{X}$ with the size of $M \times K$, where M is the number of images in a mini-batch and K is the dimension of the activation feature vector for one image.

1: **for** each input activation j from 1 to K **do**
2: $\alpha_j^1 = 0$
3: **for** each mini-batch t from 1 to T **do**
4: Feed-forward to the FC layer;
5: Select active features by boosting and calculate weights $\boldsymbol{\alpha}^t$ based on the standard AdaBoost;
6: Update the incremental strong classifier as Eq. 2.6;
7: Calculate the overall loss of IB-CNN as Eq. 2.8;
8: Backpropagate the loss based on Eq. 2.9 and Eq. 2.10;
9: Continue backpropagation to lower layers.

Substituting Eq. 2.1 into Eq. 2.5, we have

$$H_I^t(\mathbf{x}_i) = \sum_{j=1}^{K} \frac{(t-1)\alpha_j^{t-1} + \alpha_j^t}{t} h^t(x_{ij}; \lambda_j) \tag{2.6}$$

where α_j^t is calculated in the t^{th} iteration by boosting. As shown in Fig. 2.3, $h^{t-1}(\cdot)$ has been updated to $h^t(\cdot)$ by updating the threshold λ_j^{t-1} to λ_j^t. If the j^{th} weak classifier was not selected before, λ_j^t is estimated in the t^{th} iteration by boosting. Otherwise, λ_j^t will be updated from the previous iteration after backpropagation as

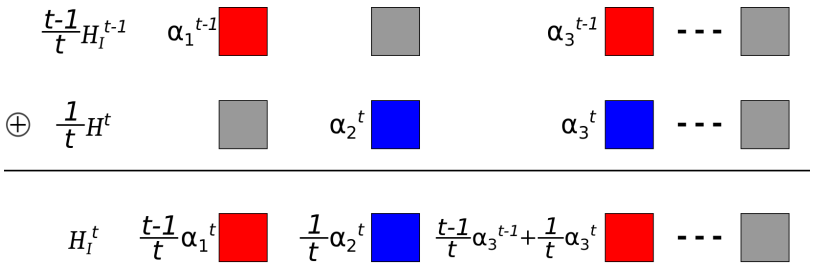

Figure 2.3 An illustration of constructing the incremental strong classifier. Squares represent neuron activations. The gray nodes are inactive; while the blue and red nodes are active nodes selected in the current iteration and previous iterations, respectively.

follows:

$$\lambda_j^t = \lambda_j^{t-1} - \gamma \nabla \frac{\partial \varepsilon^{H_I}}{\partial \lambda_j^{t-1}} \tag{2.7}$$

where γ is the learning rate.

Then, the incremental strong classifier H_I^t is updated over time. As illustrated in Figure 2.3, if a neuron is activated in the current iteration, the corresponding weight will increase; otherwise, it will decrease. The summation of weights of all weak classifiers will be normalized to 1. Hence, the weak classifiers selected most of the time, i.e., effective for most of mini-batches, will have higher weights. Therefore, the overall loss of IB-CNN is calculated as

$$\varepsilon^{IB} = \beta \varepsilon_{strong}^{IB} + (1-\beta)\varepsilon_{weak} \tag{2.8}$$

where $\varepsilon_{strong}^{IB} = \frac{1}{M}\sum_{i=1}^{M}(H_I(\mathbf{x}_i) - y_i)^2$.

Compared to the B-CNN, the IB-CNN exploits the information from all mini-batches. For testing, IB-CNN uses the incremental strong classifier, while the B-CNN employs the strong classifier learned from the last iteration.

2.2.4 IB-CNN Fine-tuning

A stochastic gradient decent method is utilized for fine-tuning the IB-CNN, i.e., updating IB-CNN parameters, by minimizing the loss in Eq. 2.8. The decent directions for x_{ij} and λ_j can be calculated as follows:

$$\frac{\partial \varepsilon^{IB}}{\partial x_{ij}} = \beta \frac{\partial \varepsilon^{IB}_{strong}}{\partial H_I(\mathbf{x}_i)} \frac{\partial H_I(\mathbf{x}_i)}{\partial x_{ij}} + (1-\beta) \frac{\partial \varepsilon^{IB}_{weak}}{\partial h(x_{ij};\lambda_j)} \frac{\partial h(x_{ij};\lambda_j)}{\partial x_{ij}} \tag{2.9}$$

$$\frac{\partial \varepsilon^{IB}}{\partial \lambda_j} = \sum_{i=1}^{M} \beta \frac{\partial \varepsilon^{IB}_{strong}}{\partial H_I(\mathbf{x}_i)} \frac{\partial H_I(\mathbf{x}_i)}{\partial \lambda_j} + (1-\beta) \sum_{i=1}^{M} \frac{\partial \varepsilon^{IB}_{weak}}{\partial h(x_{ij};\lambda_j)} \frac{\partial h(x_{ij};\lambda_j)}{\partial \lambda_j} \tag{2.10}$$

where $\frac{\partial \varepsilon^{IB}}{\partial x_{ij}}$ and $\frac{\partial \varepsilon^{IB}}{\partial \lambda_j}$ are only calculated for the active nodes of incremental boosting (the red and blue nodes in Figure 2.3). $\frac{\partial \varepsilon^{IB}}{\partial x_{ij}}$ can be further backpropagated to the lower FC layers and convolutional layers. The incremental boosting algorithm for the IB-CNN is summarized in Algorithm 1.

2.3 Experiments

2.3.1 Pre-Processing

Face alignment and facial landmark detection: Face alignment is conducted to reduce variation in face scale and in-plane rotation across different facial images. Specifically, 66 landmarks are detected using a state-of-the-art face alignment method, Discriminative Response Map Fitting (DRMF) [3]. The face regions are then aligned based on three fiducial points: the centers of the two eyes and the mouth, and scaled to a size of 128×96.

Face pose correction: In order to alleviate face pose variations, especially out-of-plane rotations, the face images are further warped to a frontal view based on landmarks, which are less affected by facial expressions including landmarks along the facial contour, two eye centers, the nose tip, and the mouth center as the red dots shown in Fig. 2.4(a). A total of 23 landmarks are selected as control points for

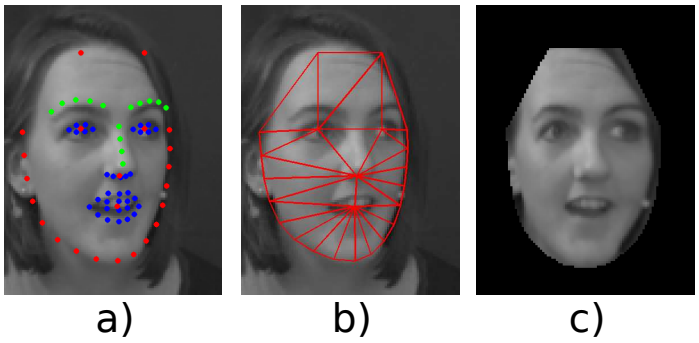

Figure 2.4 Illustration of face frontalization. (a) 23 facial landmarks (red dots) are used for face frontalization. The eye centers, mouth center, nose tip, and the two uppermost points are estimated from the landmarks (blue dots). (b) Triangularized face mesh. c) Frontalized face region enclosed in the face mesh.

face triangulation and warping as shown in Fig. 2.4(b) and (c). Specifically, the eye centers, mouth center, nose tip, and the two upper most points on the forehead are estimated from the blue landmarks detected by DRMF. The green landmarks are not used in face frontalization since they are highly sensitive to muscular movements caused by AUs. A frontalized face region is then generated from the warped face and cropped to eliminate the irrelevant regions such as the neck, hair, and background as shown in Fig.2.4(c).

Time sequence normalization is used to reduce identity-related information and highlight appearance and geometrical changes due to activation of AUs. Particularly, each image is normalized based on the mean and the standard deviation calculated from a short video sequence containing at least 800 continuous frames at a frame rate of 30fps [1].

2.3.2 CNN Implementation Details

The proposed IB-CNN is implemented based on a modification of cifar10_quick in Caffe [23]. As illustrated in Figure 2.1, the preprocessed facial images are fed into the

[1]Psychological studies show that each AU ranges from 48 frames to 800 frames at a frame rate of 30fps [55].

network as input. The IB-CNN consists of three stacked convolutional layers with activation functions, two maxpooling layers, an FC layer, and the proposed IB layer to predict the AU label. Specifically, the first two convolutional layers have 32 filters with a size of 5×5 and a stride of 1. Then, the output feature maps are sent to a rectified layer followed by the maxpooling layer with a downsampling stride of 3. The last convolutional layer has 64 filters with a size of 5×5, and the output 9×5 feature maps are fed into an FC layer with 128 nodes. The outputs of the FC layer are sent to the proposed IB layer. The stochastic gradient descent, with a momentum of 0.9 and a mini-batch size of 100, is used for training the CNN for each target AU.

2.3.3 Experimental Results

To demonstrate effectiveness of the proposed *IB-CNN*, two baseline methods are employed for comparison. The first method, denoted as *CNN*, is a traditional CNN model with a sigmoid cross entropy decision layer. The second method, denoted as *B-CNN*, is the boosting CNN described in Section 2.2.2. Both *CNN* and *B-CNN* have the same architecture as the *IB-CNN* with different decision layers.

Performance evaluation on the SEMAINE database: All the models compared were trained on the training set and evaluated on the validation set. The training-testing process was repeated 5 times. The mean and standard deviation of F1 score and two-alternative forced choice (2AFC) score are calculated from the 5 runs for each target AU. As shown in Table 2.1, the proposed *IB-CNN* outperforms the traditional *CNN* in term of the average F1 score (0.416 vs 0.347) and the average 2AFC score (0.775 vs 0.735). Not surprisingly, *IB-CNN* also beats *B-CNN* obviously: the average F1 score increases from 0.310 (*B-CNN*) to 0.416 (*IB-CNN*) and the average 2AFC score increases from 0.673 (*B-CNN*) to 0.775 (*IB-CNN*), thanks to incremental learning over time. In addition, *IB-CNN* considering both strong and weak classifier losses outperforms the one with only strong-classifier loss, denoted as

IB-CNN-S. Note that, *IB-CNN* achieves a significant improvement for recognizing AU28 (Lips suck), which has the least number of occurrences (around 1.25% positive samples) in the training set, from 0.280 (*CNN*) and 0.144 (*B-CNN*) to 0.490 (*IB-CNN*) in terms of F1 score. The performance of *B-CNN* is the worst for infrequent AUs due to the limited positive samples in each mini-batch. In contrast, the proposed *IB-CNN* improves CNN learning significantly with limited training data.

Table 2.1 Performance comparison of *CNN*, *B-CNN*, *IB-CNN-S*, and *IB-CNN* on the SEMAINE database in terms of F1 and 2AFC. The format is mean±std. PPos: percentage of positive samples in the training set.

AUs	PPos	**CNN**		**B-CNN**		**IB-CNN-S**		**IB-CNN**	
		F1	2AFC	F1	2AFC	F1	2AFC	F1	2AFC
AU2	13.5%	0.314±0.065	0.715±0.076	0.241±0.073	0.646±0.060	**0.414**±0.016	0.812±0.010	0.410±0.024	**0.820**±0.009
AU12	17.6%	0.508±0.023	0.751±0.009	**0.555**±0.007	0.746±0.013	0.549±0.016	0.773±0.007	0.539±0.013	**0.777**±0.005
AU17	1.9%	**0.288**±0.020	0.767±0.014	0.204±0.048	0.719±0.036	0.248±0.048	0.767±0.011	0.248±0.007	**0.777**±0.012
AU25	17.7%	0.358±0.033	0.635±0.011	**0.407**±0.006	0.618±0.011	0.378±0.009	0.638±0.011	0.401±0.014	**0.638**±0.003
AU28	1.25%	0.280±0.111	0.840±0.076	0.144±0.092	0.639±0.195	0.483±0.069	0.898±0.006	**0.490**±0.078	**0.904**±0.011
AU45	19.7%	0.333±0.036	0.702±0.022	0.311±0.016	0.668±0.019	**0.401**±0.009	**0.738**±0.010	0.398±0.005	0.734±0.005
AVG	-	0.347±0.026	0.735±0.014	0.310±0.015	0.673±0.028	0.412±0.018	0.771±0.003	**0.416**±0.018	**0.775**±0.004

Performance evaluation on the DISFA database: A 9-fold cross-validation strategy is employed for the DISFA database, where 8 subsets of 24 subjects were utilized for training and the remaining one subset of 3 subjects for testing. For each fold, the training-testing process was repeated 5 times. The mean and standard deviation of the F1 score and the 2AFC score are calculated from the 5×9 runs for each target AU and reported in Table 2.2. As shown in Table 2.2, the proposed *IB-CNN* improves the performance from 0.405 (*CNN*) and 0.398 (*B-CNN*) to 0.457 (*IB-CNN*) in terms of the average F1 score and from 0.780 (*CNN*) and 0.815 (*B-CNN*) to 0.823 (*IB-CNN*) in terms of 2AFC score. Similar to the results on the SEMAINE database, the performance improvement of the infrequent AUs is more impressive. AU5 (upper lid raiser) has the least number of occurrences, i.e., 2.09% positive samples, in the DISFA database. The recognition performance increases

Table 2.2 Performance comparison of *CNN*, *B-CNN*, and *IB-CNN* on the DISFA database in terms of F1 score and 2AFC score. The format is mean±std. PPos: percentage of positive samples in the whole database.

AUs	PPos	CNN		B-CNN		IB-CNN	
		F1	2AFC	F1	2AFC	F1	2AFC
AU1	6.71%	0.257±0.200	0.724±0.116	0.259±0.150	**0.780**±0.079	**0.327**±0.204	0.773±0.119
AU2	5.63%	0.346±0.226	0.769±0.119	0.333±0.197	0.835±0.085	**0.394**±0.219	**0.849**±0.073
AU4	18.8%	0.515±0.208	0.820±0.116	0.446±0.186	0.793±0.083	**0.586**±0.104	**0.886**±0.060
AU5	2.09%	0.195±0.129	0.780±0.154	0.184±0.114	0.749±0.279	**0.312**±0.153	**0.887**±0.076
AU6	14.9%	0.619±0.072	0.896±0.042	0.596±0.086	0.906±0.040	**0.624**±0.069	**0.917**±0.026
AU9	5.45%	0.340±0.131	0.859±0.081	0.331±0.115	0.895±0.057	**0.385**±0.137	**0.900**±0.057
AU12	23.5%	0.718±0.063	0.943±0.028	0.686±0.083	0.913±0.030	**0.778**±0.047	**0.953**±0.020
AU15	6.01%	0.174±0.132	0.586±0.174	**0.224**±0.120	**0.753**±0.091	0.135±0.122	0.511±0.226
AU17	9.88%	0.281±0.154	0.678±0.125	0.330±0.132	**0.763**±0.086	**0.376**±0.222	0.742±0.148
AU20	3.46%	0.134±0.113	0.604±0.155	**0.184**±0.101	**0.757**±0.083	0.126±0.069	0.628±0.151
AU25	35.2%	0.716±0.111	0.890±0.064	0.670±0.064	0.844±0.049	**0.822**±0.076	**0.922**±0.063
AU26	19.1%	0.563±0.152	0.810±0.073	0.507±0.131	0.797±0.054	**0.578**±0.155	**0.876**±0.039
AVG	-	0.405±0.055	0.780±0.036	0.398±0.059	0.815±0.031	**0.457**±0.067	**0.823**±0.031

Table 2.3 Performance comparison with the state-of-the-art methods on four benchmark databases in terms of common metrics. ACC: Average classification rate.

CK		**SEMAINE**		**BP4D**		**DISFA**		
Methods	**ACC**	**Methods**	**F1**	**Methods**	**F1**	**Methods**	**2AFC**	**ACC**
AAM [40]	0.955	LGBP [66]	0.351	LGBP [66]	0.580	Gabor [43]	N/A	0.857
Gabor+DBN [65]	0.933	CNN [17]	0.341	CNN [17]	0.522	BGCS [59]	N/A	0.868
LBP [18]	0.949	DLA-SIFT [72]	0.435	DLA-SIFT [72]	0.591	LPQ [24]	0.810	N/A
						ML-CNN [15]	0.757	0.846
CNN (baseline)	0.937	CNN (baseline)	0.347	CNN (baseline)	0.510	CNN (baseline)	0.780	0.839
IB-CNN	0.951	**IB-CNN**	0.416	**IB-CNN**	0.578	**IB-CNN**	0.825	0.858

from 0.195 (*CNN*) and 0.184 (*B-CNN*) to 0.312 (*IB-CNN*) in terms of the average F1 score.

Comparison with the State-of-the-Art methods: We further compare the proposed IB-CNN with the state-of-the-art methods, especially the CNN-based methods, evaluated on the four benchmark databases using the metrics that are common in those papers [2]. As shown in Tables 2.3, the performance of IB-CNN is comparable with the state-of-the-art methods and more importantly, outperforms the CNN-based methods.

[2]Since the testing sets of the SEMAINE and BP4D database are not available, the IB-CNN is compared with the method reported on the validation sets.

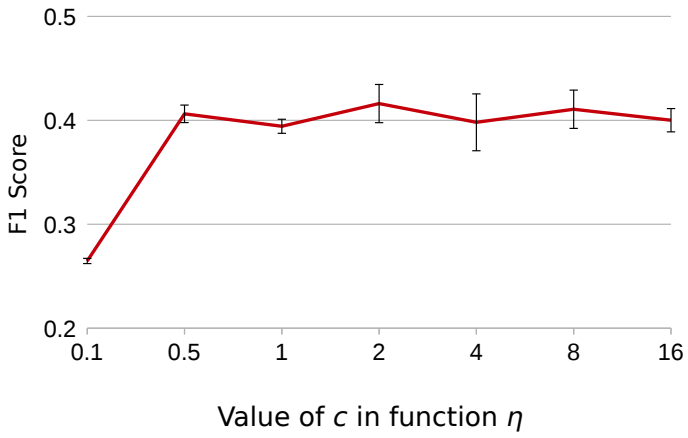

Figure 2.5 Recognition performance versus the choice of η.

2.3.4 Data Analysis

Data analysis of the parameter η: The value of η can affect the slope of the simulated $sign(\cdot)$ function and consequently, the gradient and optimization process. When η is smaller than 0.5, the simulation is more similar to the real $sign(\cdot)$, but the derivative is near zero for most of the input data, which can cause slow convergence or divergence. An experiment was conducted to analyze the influence of $\eta = \frac{\sigma}{c}$ in Eq. 2.1. Specifically, an average F1 score is calculated from all AUs in the SEMAINE database while varying the value of c. As illustrated in Figure 2.5, the recognition performance in terms of the average F1 score is robust to the choice of η when c ranges from 0.5 to 16. In our experiment, η is set to half of the standard deviation $\frac{\sigma}{2}$, empirically.

Data Analysis of the number of input neurons in the IB layer: Selecting an exact number of nodes for the hidden layers remains an open question. An experiment was conducted to demonstrate that the proposed IB-CNN is insensitive to the number of input neurons. Specifically, a set of IB-CNNs, with the number of input neurons of 64, 128, 256, 512, 1042, and 2048, were trained and tested on the SEMAINE database. For each IB-CNN, the average F1 score is computed over 5 runs

for each AU. As shown in Fig. 2.6, the *B-CNN* and especially, the proposed IB-CNN is more robust to the number of input neurons compared to the traditional *CNN* since a small set of neurons are active in contrast to the FC layer in the traditional CNN.

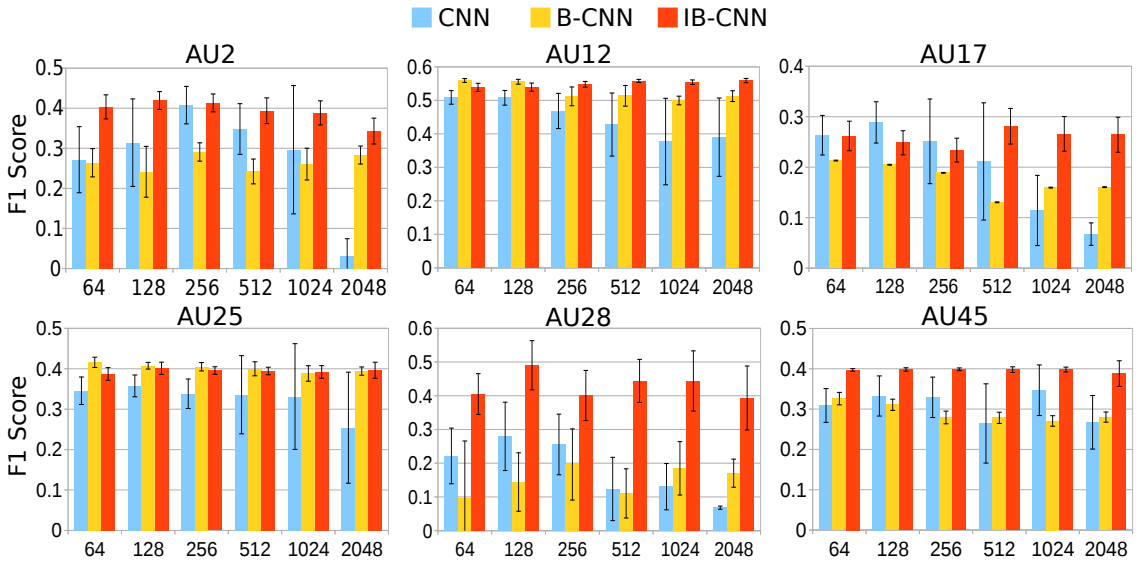

Figure 2.6 Recognition performance versus the number of input neurons in the IB layer.

Data analysis of learning rate γ: Another issue in CNNs is the choice of

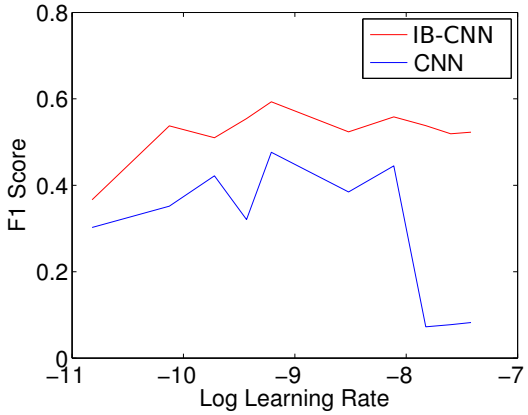

Figure 2.7 Recognition performance versus the learning rate γ.

the learning rate γ. The performance of the IB-CNN at different learning rates is depicted in Figure 2.7 in terms of the average F1 score calculated from all AUs on the SEMAINE database. Compared to the traditional *CNN*, the proposed IB-CNN is less sensitive to the value of the learning rate.

2.4 Chapter Summary

In this chapter, a novel IB-CNN was proposed to integrate boosting classification into a CNN for the application of AU recognition. To deal with limited positive samples in a mini-batch, an incremental boosting algorithm was developed to accumulate information from multiple batches over time. A novel loss function that accounts for errors from both the incremental strong classifier and individual weak classifiers is proposed to fine-tune the IB-CNN. Experimental results on four benchmark AU databases have demonstrated that the IB-CNN achieves significant improvement over the traditional CNN, as well as the state-of-the-art CNN-based methods for AU recognition. Furthermore, the IB-CNN is more effective in recognizing infrequent AUs with limited training data. The IB-CNN is a general machine learning method and can be adapted to other learning tasks, especially those with limited training data. In the future, we plan to extend it to multitask learning by replacing the binary classifier with a multiclass boosting classifier.

Chapter 3

Adaptive Convolutional Filter Size

3.1 Motivation

In CNNs, the size of the convolution filters determines the size of receptive field where information is extracted. CNN-based methods employ predefined and fixed filter sizes in each convolutional layer, which is called the *traditional CNN* hereafter. In general, larger filter sizes are employed in the lower convolutional layers, whereas smaller filter sizes are used in the upper layers [31, 8]. However, the fixed filter sizes are not necessarily optimal for all applications/tasks as well as for different input image sizes. Specifically, different AUs cause facial appearance changes over various regions at different scales and therefore, may prefer different filter sizes. For example, *long* and deep nasolabial furrows are important for recognizing AU10 (upper lip raiser), while *short* "wrinkles in the skin above and below the lips" and small bulges below the lower lip are cues for recognizing AU23 (lip tightener) [11].

Given a predefined input image size, the best filter size is often selected experimentally or by visualization [75] for each convolutional layer. For example, Kim et al. [29], who achieved the best expression recognition performance on the test set of EmotiW2015 challenge [9], experimentally selected the best filter sizes for the three convolutional layers. *However, with CNNs becoming deeper and deeper [60, 19], it is impractical to search for the best filter size by exhaustive search, due to the highly expensive training cost.*

In this work, we propose a novel and feasible solution in a CNN framework to automatically learn the filter sizes for all convolutional layers simultaneously from the training data along with learning the convolution filters. In particular, we proposed an Optimized Filter Size CNN (OFS-CNN), where the optimal filter size of each convolutional layer is estimated iteratively using stochastic gradient descent (SGD) during the *backpropagation process*. As illustrated in Figure. 3.1, the filter size k of a convolutional layer, which is a constant in the traditional CNNs, is defined as a continuous variable in the OFS-CNN. During backpropagation, the filter size k will

be updated, e.g., decreased when the partial derivative of CNN loss with respect to the filter size is positive, i.e., $\frac{\partial L}{\partial k} > 0$, and vice versa.

In this work, a forward-backward propagation algorithm is developed to estimate the filter size iteratively. To facilitate the convolution operation with a continuous filter size, *upper-bound* and *lower-bound* filters with integer-sizes are defined. In the *forward process*, an activation resulted from a convolution operation with a continuous filter size can be calculated as the interpolation of the activations using the upper-bound and lower-bound filters. Furthermore, we show that only one convolution operation is needed with the upper-bound and lower-bound filters. Therefore, the proposed OFS-CNN has similar computation complexity as the traditional CNNs in the forward process as well as in the testing process. During *backpropagation*, the partial derivative of the activation with respect to the filter size k is defined, from which $\frac{\partial L}{\partial k}$ can be calculated. With a change in the filter size k, the filter sizes of the upper-bound or lower-bound filters may be updated via a *transformation operation* proposed in this work.

3.2 Methodology

3.2.1 A Brief Review of CNNs

A CNN consists of a stack of layers such as convolutional layers, pooling layers, rectification layers, fully connected (FC) layers, and loss layers. These layers transform the input data to highly nonlinear representations. Convolutional layers are used to perform convolution on input images or feature maps from the previous layer with filters. Generally, the first convolutional layer is used to extract low-level image features such as edges; while the upper layers can extract complex and task-related features.

Given an input image/feature map denoted by $\mathbf{x}$, an activation at the i^{th} row and the j^{th} column, denoted by y_{ij}, in a convolutional layer can be calculated using the convolution operation by computing the inner product of the filter and the input as

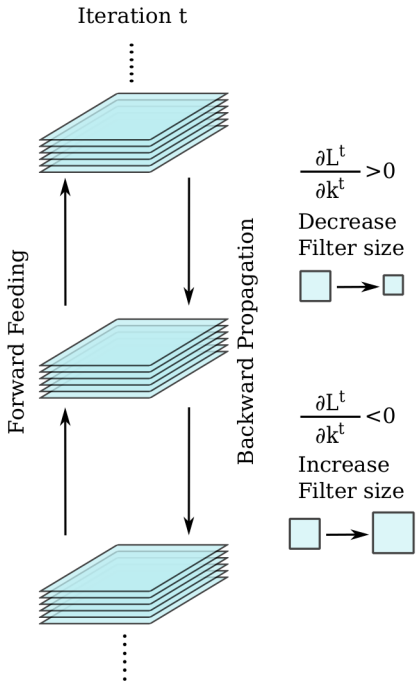

Figure 3.1 The overview of the proposed method to optimize the convolutional filter size with the loss backpropagation at iteration t. $\frac{\partial L^t}{\partial k^t}$ is the partial derivative of loss with respect to filter size k^t.

follows:

$$y_{ij}(k) = \mathbf{w}(k)^\top \mathbf{x}_{ij}(k) + b_{ij} \tag{3.1}$$

where $\mathbf{w}(k)$ is a convolution filter with the filter size $k \times k$; $\mathbf{x}_{ij}(k)$ denotes the input with a $k \times k$ receptive field centered at the i^{th} row and the j^{th} column; and b_{ij} is a bias. Traditionally, the filter size k is a predefined integer and fixed throughout the training/testing process. *In this work, $k \in \mathbb{R}^+$ is defined as a continuous variable that can be learned and optimized during CNN training.*

3.2.2 Forward Processing of the OFS-CNN

In the forward process, convolution operations are conducted to calculate activations using learned filters as in Eq. 3.1. However, the convolution operation can only be performed with integral size filters in the CNN.

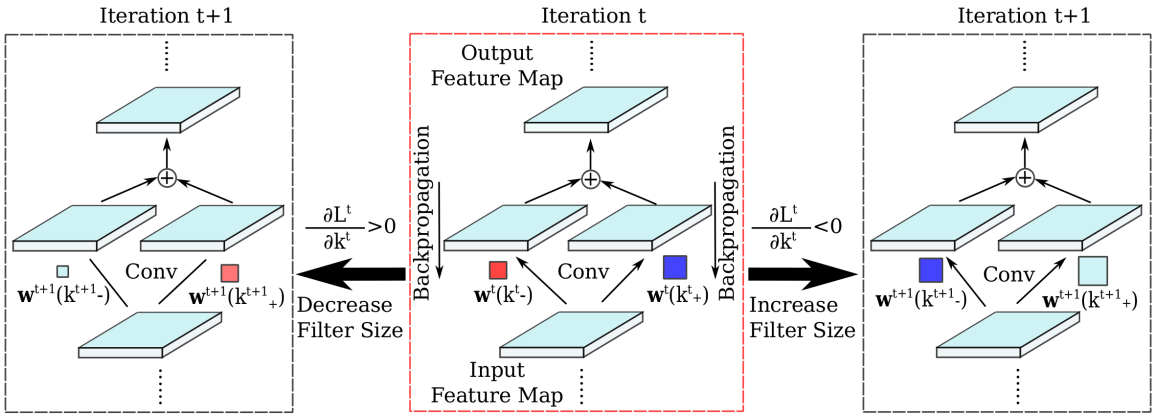

Figure 3.2 The strategy to optimize the convolutional filter size based on gradient descent method. $\frac{\partial L^t}{\partial k^t}$ is the partial derivative of loss with respect to filter size k^t at iteration t. $\mathbf{w}^t(k_+^t)$ and $\mathbf{w}^t(k_-^t)$ are used to approximate the derivative as equation 3.15

Upper-bound and lower-bound filters: In order to build the relationship between the activation y_{ij} and the continuous filter size k, we first define an *upper-bound filter* denoted by $\mathbf{w}(k_+)$ and a *lower-bound filter* denoted by $\mathbf{w}(k_-)$. Specifically, k_+ is the upper-bound filter size and is the smallest odd number that is bigger than k; while k_- is the lower-bound filter size and is the largest odd number that is less than or equal to k. k_+ and k_- can be calculated as

$$k_+ = \lfloor \frac{k+1}{2} \rfloor * 2 + 1, \quad k_- = \lfloor \frac{k+1}{2} \rfloor * 2 - 1 \tag{3.2}$$

Then, the activation $y_{ij}(k)$ can be defined as the linear interpolation of the activations of the upper-bound and lower-bound filters denoted by $y_{ij}(k_-)$ and $y_{ij}(k_+)$, respectively:

$$y_{ij}(k) = \alpha y_{ij}(k_+) + (1-\alpha) y_{ij}(k_-) \tag{3.3}$$

$$\alpha = \frac{(k-k_-)}{2} \tag{3.4}$$

where $y_{ij}(k_+)$ and $y_{ij}(k_-)$ are calculated as in Eq. 3.1 with the same bias, but with the upper-bound and lower-bound filters ($\mathbf{w}(k_+)$ and $\mathbf{w}(k_-)$), respectively. α is the linear interpolation weight.

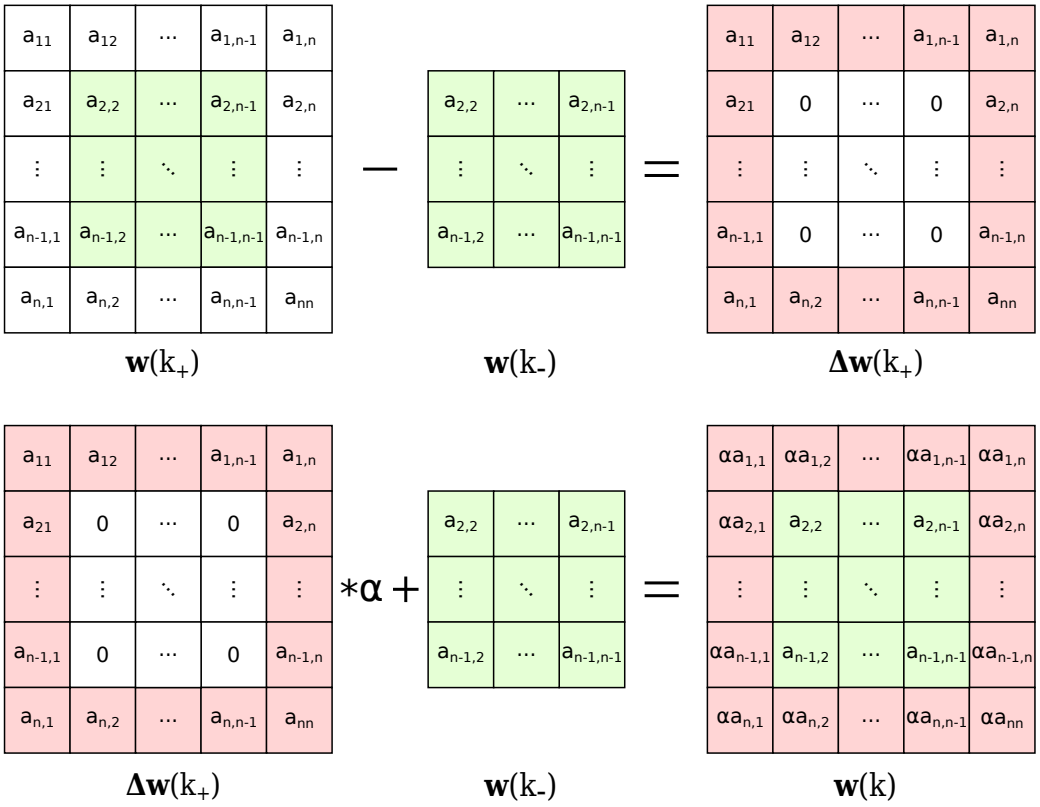

Figure 3.3 An illustrative definition of a filter with a continuous filter size $k \in \mathbb{R}^+$. $\mathbf{w}(k_+)$ and $\mathbf{w}(k_-)$ are the upper-bound and lower-bound filters, respectively, and share the same elements in the green region. The pink region $\triangle\mathbf{w}(k_+)$ denotes the difference between the upper-bound and lower-bound filters and has a ring shape with zeros inside. α defined as in Eq. 3.4 is the linear interpolation weight associated with the upper-bound filter $\mathbf{w}(k_+)$. $\mathbf{w}(k)$ is a weight-related filter with a continuous filter size k.

Remark 1. *A cubic interpolation can also be used to build the relationship between the activation y_{ij} and the continuous variable k. However, it requires a higher computational complexity and needs at least three points; while the linear interpolation only needs two points k_- and k_+.*

Remark 2. *The filter size k is actually a weight-related filter size in the interval $[k_-, k_+)$. Based on Eq. 3.4, it can be calculated as:*

$$k = k_- + 2\alpha \tag{3.5}$$

Convolution with a continuous filter size: As in Remark 2, we can explicitly define the filter $\mathbf{w}(k)$ with a continuous size k. As shown in Fig. 3.3, the upper-bound and lower-bound filters are defined to share the same coefficients in the region with green color and to differ by the pink boundary denoted by $\triangle\mathbf{w}(k_+)$. Let $\triangle\mathbf{w}(k_+) = \mathbf{w}(k_+) - \mathbf{w}(k_-)$ be the ring boundary with zeros inside as shown in Fig. 3.3, then the filter $\mathbf{w}(k)$ with a continuous size k can be defined as follows:

$$\mathbf{w}(k) = \alpha \triangle \mathbf{w}(k_+) + \mathbf{w}(k_-), \tag{3.6}$$

Remark 3. *In Eq. 3.6, $\boldsymbol{w}(k)$ and $\boldsymbol{w}(k_-)$ have an actual filter size of k_+; while $\boldsymbol{w}(k_-)$ is zero-padded.*

Lemma 1. *Given the definition of the filter $\boldsymbol{w}(k)$ as in Eq. 3.6, the activation $y_{ij}(k)$ in Eq. 3.3 can be simplified as:*

$$y_{ij}(k) = \boldsymbol{w}(k)^\top \boldsymbol{x}_{ij}(k_+) + b_{ij} \tag{3.7}$$

Proof. Eq. 3.7 can be deduced step by step from Eq. 3.3 as follows:

$$\begin{aligned} y_{ij}(k) =& \alpha y_{ij}(k_+) + (1-\alpha) y_{ij}(k_-) \\ =& \alpha \mathbf{w}(k_+)^\top \mathbf{x}(k_+) + (1-\alpha)\mathbf{w}(k_-)^\top \mathbf{x}(k_-) + b_{ij} \end{aligned} \tag{3.8}$$

After padding zeros for $\mathbf{w}(k_-)$, $\mathbf{w}(k_-)^\top \mathbf{x}(k_-)$ is equivalent to $\mathbf{w}(k_-)^\top \mathbf{x}(k_+)$. Then, Eq. 3.8 can be simplified as follows:

$$\begin{aligned} y_{ij}(k) =& \alpha \mathbf{w}(k_+)^\top \mathbf{x}(k_+) + (1-\alpha)\mathbf{w}(k_-)^\top \mathbf{x}(k_+) + b_{ij} \\ =& \left[\alpha \mathbf{w}(k_+)^\top + (1-\alpha)\mathbf{w}(k_-)^\top\right] \mathbf{x}(k_+) + b_{ij} \\ =& \left[\alpha \triangle \mathbf{w}(k^+)^\top + \mathbf{w}(k_-)^\top\right] \mathbf{x}(k_+) + b_{ij} \end{aligned} \tag{3.9}$$

By substituting Eq. 3.6 into Eq. 3.9, we have

$$y_{ij}(k) = \mathbf{w}(k)^\top \mathbf{x}_{ij}(k_+) + b_{ij} \tag{3.10}$$

Thus, the activation of $y_{ij}(k)$ can be simplified as Eq. 3.7. □

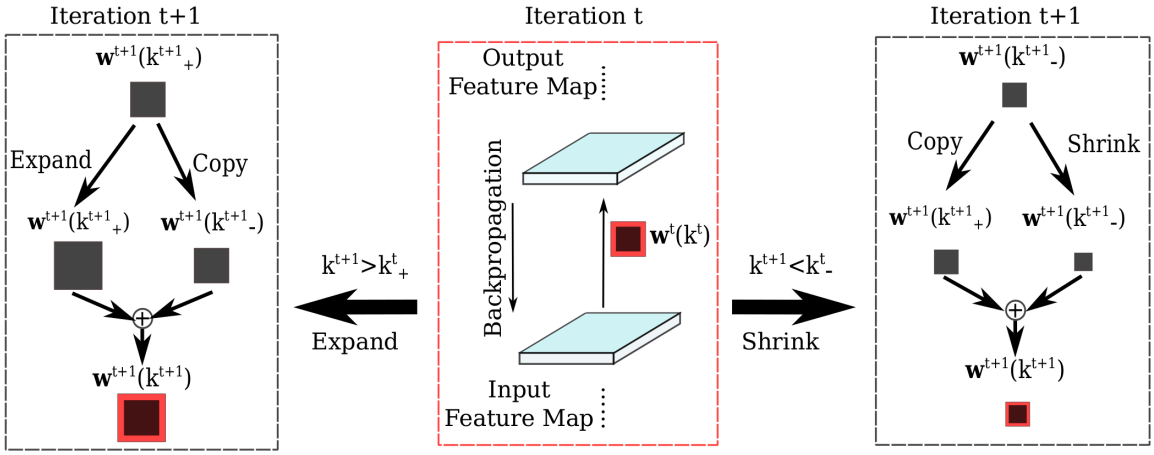

Figure 3.4 When the filter size k is updated during backpropagation, it may be out of the interval $[k_-^t, k_+^t)$. In this case, transformation operations are needed to update the sizes of the upper-bound and lower-bound filters after updating their coefficients. Specifically, an expanding operation is employed to increase the sizes of both upper-bound and lower-bound filters; whereas a shrinking operation is used to decrease the filter sizes.

Remark 4. *According to Eq. 3.7, only one convolution operation needs to be performed to calculate each activation $y_{ij}(k)$. Therefore, the time complexity does not increase compared with the traditional CNN in the forward training process as well as in the testing process.*

3.2.3 Backward propagation of the OFS-CNN

Optimizing filter size in the OFS-CNN

Calculating the partial derivative: Since the relationship between the activation and the filter size has been defined as in Eq. 3.3, the partial derivative of the activation y_{ij} with respect to the filter size can be calculated based on the derivative definition as follows:

$$\frac{\partial y_{ij}(k)}{\partial k} = \lim_{\triangle k \to 0} \frac{y_{ij}(k+\triangle k) - y_{ij}(k-\triangle k)}{2 \triangle k} \tag{3.11}$$

When $k+\triangle k$ and $k-\triangle k$ are in the interval $[k_-, k_+)$, the derivative of each point $\frac{\partial y_{ij}(k)}{\partial k}$ is equal to the gradient of the line because of the linear interpolation. Hence,

the partial derivative can be calculated as follows:

$$\frac{\partial y_{ij}(k)}{\partial k} = \frac{y_{ij}(k_+) - y_{ij}(k_-)}{k_+ - k_-} \tag{3.12}$$

Substituting Eq. 3.1 into Eq. 3.12, we have

$$\frac{\partial y_{ij}(k)}{\partial k} = \frac{\mathbf{w}(k_+)^\top \mathbf{x}_{ij}(k_+) - \mathbf{w}(k_-)^\top \mathbf{x}_{ij}(k_-)}{k_+ - k_-} \tag{3.13}$$

By padding zeros for $\mathbf{w}(k_-)$, we can simplify Eq. 3.13 as

$$\begin{aligned}\frac{\partial y_{ij}(k)}{\partial k} =& \frac{\mathbf{w}(k_+)^\top \mathbf{x}_{ij}(k_+) - \mathbf{w}(k_-)^\top \mathbf{x}_{ij}(k_+)}{k_+ - k_-} \\ =& \frac{\left[\mathbf{w}(k_+)^\top - \mathbf{w}(k_-)^\top\right] \mathbf{x}_{ij}(k_+)}{k_+ - k_-} \\ =& \frac{\triangle \mathbf{w}(k_+)^\top \mathbf{x}_{ij}(k_+)}{k_+ - k_-}\end{aligned} \tag{3.14}$$

Based on Eq. 3.14, the partial derivative of the loss L with respect to k can be calculated as follows with chain rule:

$$\frac{\partial L}{\partial k} = \sum_{i,j} \frac{\partial L}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial k} \tag{3.15}$$

Updating the filter size: Given the partial derivative of the loss L with respect to k, the filter size k can be updated iteratively with the SGD strategy for the $(t+1)^{th}$ iteration as follows:

$$k^{t+1} = k^t - \gamma(\frac{\partial L^t}{\partial k^t} + \eta k^t) \tag{3.16}$$

where γ is the learning rate. η is the penalty parameter for bigger filter size. The experiments will show the different initialization of the filter sizes, and there is penaltiy effect when initializing with bigger filter size.

Updating convolution filters $\mathbf{w(k)}$

Updating the upper-bound and lower-bound filters: Since the lower-bound filter $\mathbf{w}^t(k_-)$ is defined as the inner part of the upper-bound filter $\mathbf{w}^t(k_+)$, we only need to perform backpropagation for the upper-bound filter $\mathbf{w}^t(k_+)$, which can be divided into two parts as $\mathbf{w}^t(k_+) = \mathbf{w}^t(k_-) + \triangle \mathbf{w}^t(k_+)$, where $\triangle \mathbf{w}^t(k_+)$ is the ring

boundary with zeros inside and $\mathbf{w}(k_-)$ is padded with zeros. Then, the forward activation function in Eq. 3.7 can be reorganized as:

$$\begin{aligned} y_{ij}^t(k^t) =& \mathbf{w}^t(k^t)^\top \mathbf{x}_{ij}^t(k_+^t) + b_{ij}^t \\ =& \left[\alpha^t \triangle \mathbf{w}^t(k_+^t)^\top + \mathbf{w}^t(k_-^t)^\top\right] \mathbf{x}^t(k_+^t) + b_{ij}^t \\ =& \alpha^t \triangle \mathbf{w}(k_+^t)^\top \triangle \mathbf{x}^t(k_+^t) + \mathbf{w}^t(k_-^t)^\top \mathbf{x}^t(k_-^t) + b_{ij}^t \end{aligned} \tag{3.17}$$

where $\triangle \mathbf{x}^t(k_+^t)$ is the ring boundary around $\mathbf{x}^t(k_-^t)$ in the input image/feature map.

Hence, the partial derivative of the activation y_{ij}^t with respect to the upper-bound filter $\mathbf{w}^t(k_+^t)$ can be calculated as follows:

$$\frac{\partial y_{ij}^t}{\partial \mathbf{w}^t(k_+^t)} = \mathbf{x}_{ij}^t(k_+^t)^\top + \alpha^t \triangle \mathbf{x}_{ij}^t(k_+^t)^\top \tag{3.18}$$

With the chain rule, the derivative of CNN loss with respect to $\mathbf{w}^t(k_+)$ can be calculated as

$$\frac{\partial L^t}{\partial \mathbf{w}^t(k_+^t)} = \sum_{i,j} \frac{\partial L^t}{\partial y_{ij}^t} \frac{\partial y_{ij}^t}{\partial \mathbf{w}^t(k_+^t)} \tag{3.19}$$

Thus, the upper-bound filter $\mathbf{w}(k_+)$ can be updated iteratively using the SGD strategy. As a result, the filter $\mathbf{w}(k)$ with a continuous size k can be updated from $\mathbf{w}(k_+)$ as in Eq. 3.6.

Transforming the upper-bound and lower-bound filters: According to Eq. 3.16, the filter size k can be continuously updated over time. As long as k^{t+1} is in the interval of $[k_-^t, k_+^t)$, the upper-bound and lower bound filters remain the same sizes as those in the t^{th} iteration, i.e., $k_-^{t+1} = k_-^t$ and $k_+^{t+1} = k_+^t$. However, as the filter size k is updated, it may become greater than k_+^t or smaller than k_-^t, i.e., k^{t+1} is outside of the interval of $[k_-^t, k_+^t)$. Consequently, both the sizes of the upper-bound and lower-bound filters should be updated. In this work, we define *transformation operations*, including *expanding* and *shrinking* to update the upper-bound and lower-bound filters to accommodate a size change.

Note that, the transformation operations are conducted after updating coefficients of the upper-bound and lower-bound filters.

Expanding: When the updated filter size is bigger than the upper-bound filter size in the previous iteration, i.e., $k^{t+1} > k_+^t$, the upper-bound and lower-bound filters $\mathbf{w}^{t+1}(k_+^{t+1})$ and $\mathbf{w}^{t+1}(k_-^{t+1})$ should be updated by an expanding operation as follows:

$$\begin{aligned}\mathbf{w}^{t+1}(k_-^{t+1}) =& \mathbf{w}^{t+1}(k_+^{t+1}) \\ \mathbf{w}^{t+1}(k_+^{t+1}) =& expand(\mathbf{w}^{t+1}(k_+^{t+1}))\end{aligned} \tag{3.20}$$

where $expand(\cdot)$ is a function to increase the filter size, particularly by padding values from the nearest neighbors of the original filter as illustrated in Figure 3.5.

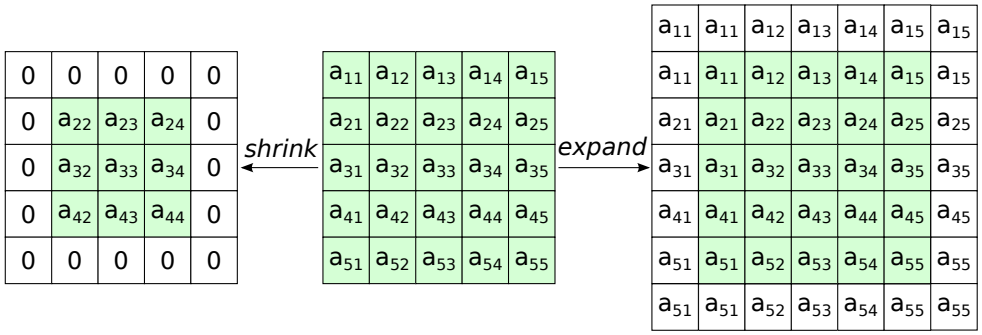

Figure 3.5 An illustration of the *shrink* and *expand* operations to change the filter size. The *shrink* operation sets zeros to the outside boundary; while the *expand* operation is to pad the outside boundary with the nearest neighbors from the original filter.

Shrinking: As opposed to the $expand(\cdot)$ function, when the filter size becomes smaller than k_-^t, the upper-bound and lower-bound filters $\mathbf{w}^{t+1}(k_-^{t+1})$ and $\mathbf{w}^{t+1}(k_+^{t+1})$ will be shrunk as follows

$$\begin{aligned}\mathbf{w}^{t+1}(k_+^{t+1}) =& \mathbf{w}^{t+1}(k_-^{t+1}) \\ \mathbf{w}^{t+1}(k_-^{t+1}) =& shrink(\mathbf{w}^{t+1}(k_-^{t+1}))\end{aligned} \tag{3.21}$$

where $shrink(\cdot)$ is a function to decrease the filter size, specifically by filling the boundary with zeros as shown in Fig. 3.5.

Remark 5. *There are alternative methods that can be used to expand or shrink the filters. For example, we have also tried to resize the filter by bicubic interpolation.*

However, the recognition performance became worse. The reason is that the filters learned in the previous iterations are distorted after scaling and thus, may fail to activate the patterns in the images. In contrast, the proposed expand and shrink functions can well preserve the learned filters.

Updating other parameters: In addition to updating the filter size k and the convolution filter $\mathbf{w}(k)$, we should also update the bias b_{ij} and the feature $\mathbf{x}_{ij}$ during backpropagation.

Based on the forward activation function as defined in Eq. 3.7, the derivative of feature activation y_{ij}^t with respect to $\mathbf{x}_{ij}^t$ can be calculated as below:

$$\frac{\partial y_{ij}^t}{\partial \mathbf{x}_{ij}^t} = \mathbf{w}^t(k^t) \tag{3.22}$$

With the chain rule, the derivative of CNN loss with respect to $\mathbf{x}_{ij}^t$ can be calculated as:

$$\frac{\partial L^t}{\partial \mathbf{x}_{ij}^t} = \frac{\partial L^t}{\partial y_{ij}^t}\frac{\partial y_{ij}^t}{\partial \mathbf{x}_{ij}^t} \tag{3.23}$$

Hence, the feature $\mathbf{x}_{ij}$ can be updated using the SGD strategy and will be further backpropagated to update the parameters in the lower layers. The backpropagation of b_{ij}^t is exactly the same as that in the traditional CNNs. The forward and backward propagation process for the proposed OFS-CNN is summarized in Algorithm 2.

3.2.4 Across-channel and within-channel filter size optimization

There are two settings for the filter size optimization: the across-channels and within-channel. For the across-channels filter size optimization, all the channels share one same filter size, which will be optimized across all channels as shown in Eq. 3.24. For the within-channel filter size optimization, each channel has a different filter size that will be optimized separately as shown in Eq. 3.25. The motivation for the within-channel model is that the different channels learn the different filter patterns

Algorithm 2 The forward-backward propagation algorithm for the OFS-CNN

Input: Input images or feature maps from the previous layer $\mathbf{x}$ and an initial filter size $k^0 \in \mathbb{R}^+$.

Initialization:

Initialize k_+^0 and k_-^0 as Eq. 3.2.

Randomly initialize the convolution filter $\mathbf{w}^0(k_+^0)$.

for iteration t from 0 to T **do**

//Forward:

$\mathbf{w}^t(k_-^t) = shrink(\mathbf{w}^t(k_+^t))$

Calculate the convolution filter $\mathbf{w}^t(k^t)$ based on Eq. 3.6

Calculate the forward activation $y_{ij}(k)$ based on Eq. 3.7 **//Backward:**

Calculate the derivative of activation with respect to k^t, $\mathbf{w}^t(k_+^t)$, and $\mathbf{x}^t$, based on Eq.3.14, Eq.3.18, and Eq. 3.22, respectively

Calculate the derivative of loss with respect to k^t, $\mathbf{w}^t(k_+^t)$, and $\mathbf{x}^t$, based on Eq.3.15, Eq.3.19, and Eq. 3.23, respectively

Update k^{t+1}, $\mathbf{w}^{t+1}(k_+^{t+1})$, and $\mathbf{x}^{t+1}$ based on SGD

Update the bias using the standard CNN backpropagation

//**Transformation:**

if $k^{t+1} > k_+^t$ **then**

$k_-^{t+1} = k_+^t$

$k_+^{t+1} = k_+^t + 2$

Expand the upper-bound and lower bound filters $\mathbf{w}^{t+1}(k_+^{t+1})$ and $\mathbf{w}^{t+1}(k_-^{t+1})$ as in Eq. 3.20

else if $k^{t+1} < k_-^t$ **then**

$k_+^{t+1} = k_-^t$

$k_-^{t+1} = k_-^t - 2$

Shrink the upper-bound and lower bound filters $\mathbf{w}^{t+1}(k_+^{t+1})$ and $\mathbf{w}^{t+1}(k_-^{t+1})$ as in Eq. 3.21

and thus need the different filter sizes. In the experiment part, we will compare the performance of across-channels and within-channel filter size optimization.

$$\frac{\partial L}{\partial k} = \sum_{i,j,c} \frac{\partial L}{\partial y_{ijc}} \frac{\partial y_{ijc}}{\partial k} \tag{3.24}$$

$$\frac{\partial L}{\partial k_c} = \sum_{i,j} \frac{\partial L}{\partial y_{ijc}} \frac{\partial y_{ijc}}{\partial k_c} \tag{3.25}$$

Table 3.1 Performance comparison of the proposed OFS-CNN and traditional CNNs with varying filter size in the first convolutional layer on the SEMAINE database [66]. In the *1-layer OFS-CNN*, the filter size is learned only for the first layer. The average converged filter size is reported for each AU, respectively. All the CNNs in comparison used the fixed filter sizes (5 and 5) for the other two layers. The results are calculated from 5 runs and formatted as mean±std in terms of the average F1 score and the 2AFC score. The underline highlights the best performance among the 4 fixed filter sizes. The **bold** highlights the best performance among all models.

AUs	F1 score					Converged Size
	CNN-Filter3	CNN-Filter5	CNN-Filter7	CNN-Filter9	1-layer OFS-CNN	
AU2	0.353±0.033	0.369±0.018	0.381±0.014	0.357±0.024	**0.412**±0.017	5.8
AU12	**0.553**±0.009	0.550±0.007	0.545±0.014	0.551±0.002	0.548±0.016	6.4
AU17	0.294±0.015	0.310±0.019	**0.322**±0.011	0.312±0.018	0.297±0.011	6.4
AU25	0.343±0.016	0.348±0.008	0.341±0.017	**0.352**±0.013	0.347±0.015	5.4
AU28	0.234±0.032	0.288±0.011	0.300±0.042	0.315±0.044	**0.360**±0.073	6.7
AU45	0.290±0.018	0.310±0.005	0.320±0.011	0.308±0.012	**0.326**±0.004	6.1
AVE	0.344±0.008	0.363±0.006	0.368±0.009	0.366±0.007	**0.382**±0.014	6.1
	2AFC					
AU2	0.766±0.024	0.798±0.007	0.799±0.019	0.784±0.023	**0.823**±0.013	5.8
AU12	0.749±0.008	0.753±0.005	0.751±0.015	**0.756**±0.005	0.745±0.012	6.4
AU17	0.814±0.011	0.811±0.013	**0.817**±0.003	0.808±0.011	0.800±0.005	6.4
AU25	0.569±0.007	0.570±0.01	0.575±0.007	**0.587**±0.012	0.585±0.010	5.4
AU28	0.831±0.014	0.822±0.01	0.823±0.01	0.825±0.006	**0.850**±0.017	6.7
AU45	0.642±0.019	0.669±0.007	0.672±0.011	0.671±0.003	**0.681**±0.009	6.1
AVE	0.728±0.006	0.737±0.004	0.740±0.007	0.739±0.003	**0.747**±0.005	6.1

3.3 Experiments

3.3.1 Pre-Processing

The steps of pre-processing include face image alignment and time sequence normalization same as in Section 2.3.1.

3.3.2 CNN Implementation Details

The proposed CNN structure for facial action unit recognitoin is similar to Section 2.3.2. There are two differences. First, the sigmoid cross entropy loss layer is used for calculating the loss. Second, all filter sizes are 5×5 in the original cifar10_quick [23] and will be used for the baseline CNN for comparison. In the OFS-CNN, all filter sizes are initialized with 4 as default, implying $\alpha^0 = 0.5$, $k_+^0 = 5$, and $k_-^0 = 3$.

For the structure of expression recognition, there are also three convolutional layers with 3×3 filter size. The first convolutional layer has 64 filters with stride 2. The second and third convolutional layers have 128 filters with stride 2. Two max-pooling layers with stride 2 are followed after the first and second convolutional layers. Two fully connected layers with 1024 neurons are followed after the last convolutional layer. The BatchNormlization, ReLU and Dropout are employed in our network. The loss layer is softmax loss. In the OFS-CNN, all filters sizes are initilized with 2×2, implying $\alpha^0 = 0.5$, $k_+^0 = 3$, and $k_-^0 = 1$.

3.3.3 Experimental Results

The proposed OFS-CNN is compared with the baseline CNN with fixed convolution filter sizes on the four AU-coded benchmark datasets and one spontaneous facial expression database. The CK database [28], The SEMAINE and the BP4D databases [66] for the FERA2015 AU recognition challenge, the DISFA database [43]. Experimental results are reported in terms of the average F1 score and 2AFC (area under ROC curve). Because of the limited subjects number for expression databases, all less than 30 in training, the experiment is evaluated on the SFEW database with more than one thousand subjects for facial expression recognition.

Exhaustive search vs filter size optimization: We will first show that the proposed OFS-CNN is capable of learning the optimal filter sizes. Specifically, baseline CNNs are designed with varying filter sizes including 3×3, 5×5, 7×7, and 9×9 in the first convolutional layer. In contrast, a *1-layer OFS-CNN* is designed where the filter size is learned only for the first layer. All the models in comparison used the fixed filter sizes (each 5×5) for the other two layers and are trained on the training partition and tested on the development partition of the SEMAINE database [66]. The results are reported in Table 3.1, which are calculated from 5 runs and formatted as mean±std. The last column lists the average filter size at the 2000^{th} iteration,

which most of the CNN models are converged in our experiments.

As shown in Table 3.1, the *1-layer OFS-CNN* not only outperforms *CNN-Filter5* with the fixed filter size 5×5, i.e., the original cifar10_quick [23] in terms of the average F1 score (0.382 vs 0.363) and the average 2AFC score (0.747 vs 0.737), but also achieves the best performance among all models compared to in terms of the average F1 score and 2AFC score. This demonstrates that the proposed OFS-CNN is superior to or at least comparable to the best CNN model obtained by exhaustive search. In addition, the learned filter size is often consistent with the best filter size obtained by exhaustive search, which is either the upper-bound or lower-bound filter size in the OFS-CNN.

The performance improvement using the *1-layer OFS-CNN* is more impressive for AU2 (outer brow raiser) and AU28 (lip suck). AU28 has the largest converged filter size of 6.7 by the *1-layer OFS-CNN*, which is consistent with the appearance changes caused by AU28: when AU28 is activated, the lips are pulled and sucked into the mouth and thus, have a long and thin "—" shape [11].

OFS-CNNs on different image resolutions: We also show that the learned filter sizes adapt well to changes in image resolutions. Specifically, experiments have been conducted to compare the proposed OFS-CNN and the baseline CNN on the BP4D database [66] with different resolutions of the input images. All the CNN models have similar CNN structure as described in Section 3.3.2. In order to accommodate the changes in the resolution, the number of nodes in the first FC layer is set to 64, 128, and 256 for resolutions of 64×48, 128×96, and 256×192, respectively, for all models in comparison. In this set of experiments, the filter sizes in all three convolutional layers are learned in the proposed OFS-CNN and the average converged filter sizes for each AU under each resolution are reported in Table 3.3.

From Tables 3.2 and 3.3, we can find that most of AUs prefer a higher image resolution to preserve subtle cues of facial appearance changes and the converged

Table 3.2 Performance comparison of the proposed OFS-CNN and the baseline CNN for varying image resolutions on the BP4D database [66] in terms of the average F1 score. The **bold** highlights the best performance among all models.

Resolution	64×48		128× 96		256×192	
Layer	CNN	OFS-CNN	CNN	OFS-CNN	CNN	OFS-CNN
AU1	0.313	0.348	0.340	0.345	0.332	**0.416**
AU2	0.277	**0.312**	0.307	0.303	0.278	0.305
AU4	0.358	0.376	0.411	**0.415**	0.324	0.391
AU6	0.693	0.723	0.721	0.729	0.676	**0.745**
AU7	0.643	0.634	0.642	**0.649**	0.504	0.628
AU10	0.726	0.739	0.718	**0.754**	0.690	0.743
AU12	0.763	0.799	0.774	0.805	0.697	**0.812**
AU14	0.517	0.532	0.552	**0.562**	0.544	0.555
AU15	0.296	0.300	0.331	**0.337**	0.323	0.326
AU17	0.550	0.542	0.561	0.563	0.540	**0.568**
AU23	0.348	0.355	0.381	0.398	0.354	**0.413**
AVE	0.499	0.515	0.522	0.533	0.478	**0.537**

Table 3.3 The average converged filter sizes for varying image resolutions on the BP4D database [66]. The **bold** highlights the filter sizes with the best performance.

Resolution	64×48			128× 96			256×192		
Layer	conv1	conv2	conv3	conv1	conv2	conv3	conv1	conv2	conv3
AU1	5.2	5.1	5.1	5.5	4.9	5.1	**6.2**	**4.9**	**4.9**
AU2	**5.2**	**5.3**	**4.9**	6.0	4.8	4.9	5.9	5.3	5.1
AU4	5.1	5.5	4.8	**5.7**	**5.8**	**5.7**	5.8	5.8	5.8
AU6	5.1	4.7	4.7	5.4	4.7	4.7	**5.7**	**4.8**	**4.8**
AU7	5.0	4.8	4.7	**5.3**	**4.6**	**4.7**	5.6	4.8	4.8
AU10	4.6	5.1	4.8	**5.5**	**4.8**	**4.8**	5.5	5.5	4.9
AU12	4.8	5.9	4.8	5.2	5.5	5.9	**5.7**	**5.5**	**5.4**
AU14	5.1	4.6	4.5	**5.3**	**4.6**	**4.6**	5.9	4.6	4.5
AU15	5.2	4.9	4.8	**5.5**	**4.8**	**4.8**	5.5	4.8	4.8
AU17	4.9	4.7	4.6	5.6	4.5	4.5	**5.3**	**4.6**	**4.5**
AU23	5.4	4.6	4.7	6.0	4.7	4.7	**5.9**	**4.8**	**4.7**
AVE	5.0	5.0	4.8	5.5	5.0	5.0	5.7	5.0	5.0

filter size increases slightly for different resolutions in the first convolutional layer. As shown in Table 3.2, the proposed OFS-CNN outperforms the baseline CNN for all image resolutions, especially for 256×192, in terms of the average F1 score. When the image resolution increases to 256×192, the receptive field covers a smaller actual area of the whole face when using the same 5×5 filter size, compared to lower resolutions. In contrast, the proposed OFS-CNN has the largest average filter size of 5.7 for conv1 (the first convolutional layer) for 256×192 and thus, can benefit from an increased receptive field because of the 7×7 upper-bound filter.

Table 3.4 Performance of the proposed OFS-CNN on the BP4D database [66] with different initialization of the kernel sizes.

Initialization	4×4	6×6	8×8
AU1	0.345	0.354	**0.364**
AU2	0.303	0.321	**0.325**
AU4	**0.415**	0.404	0.397
AU6	**0.729**	0.719	0.726
AU7	0.649	**0.651**	0.641
AU10	0.754	**0.755**	0.752
AU12	0.805	0.798	**0.810**
AU14	0.562	**0.580**	0.539
AU15	0.337	**0.340**	0.336
AU17	0.563	**0.570**	**0.570**
AU23	0.398	**0.403**	0.400
AVE	0.533	**0.536**	0.533

Table 3.5 Converged filter size at the first convolutional layer on the BP4D database [66] with different initialization of the kernel sizes.

Initialization	4×4	6×6	8×8
AU1	5.0	6.0	**7.5**
AU2	5.2	6.1	**7.3**
AU4	**5.0**	6.9	8.1
AU6	**5.0**	5.9	7.7
AU7	4.9	**5.8**	6.4
AU10	5.0	**6.4**	7.1
AU12	5.2	6.8	**7.8**
AU14	4.9	**5.8**	5.8
AU15	4.9	**5.6**	7.4
AU17	5.1	**5.9**	**6.4**
AU23	5.1	**6.6**	7.5
AVE	5.0	**6.4**	7.5

Comparison with the different filter sizes initilization: This experiments can show our poposed method is not sensitive to the different initialization of filter sizes. In this experiment, the different filter sizes are initilized with 4×4, 6×6, and 8×8. As shown in Table 3.4, The different filter size intilializations have very similar performance in term of average F1 score. For the converged filter sizes in Table 3.5, the filter size increased to 5.0 in average with 4×4 initialization, and the filter size decreased to 7.5 in average with 8×8 initilization. As shown in Eq. 3.16, when the filter size becomes bigger and bigger, the penalty will increase because of the weight decay parameter η.

Comparison with the GoogLeNet The proposed OFS-CNN model has been compared with the Inception module. In particular, the GoogLeNet with 7 inception modules is evaluated on the BP4D database. The experimental results are listed in Table 3.6.

Comparisons show that the OFS-CNN with a shallow structure (15 layers, trained in 3,000 iterations) performs noticeably better than the GoogLeNet (100 layers, trained in 20,000 iterations) in terms of average F1 score. The improvement becomes

Table 3.6 Comparison with the GoogLeNet on BP4D database in terms of F1 score.

AUs	%	GoogLeNet	OFS-CNN
AU1	23.1	**0.369**	0.345
AU2	17.9	0.267	**0.303**
AU4	22.7	**0.498**	0.415
AU6	46.0	**0.746**	0.729
AU7	52.6	**0.657**	0.649
AU10	59.6	**0.768**	0.754
AU12	55.8	**0.836**	0.805
AU14	52.1	0.503	**0.562**
AU15	18.0	0.325	**0.337**
AU17	32.6	0.511	**0.563**
AU23	17.0	0.376	**0.398**
AVE	-	0.531	**0.533**

more substantial for the AUs with a lower occurrence rate such as AU2 (17.9%). The GoogLeNet is much more complex compared to our approach and thus, demands more training data. Note that the proposed OFS-CNN runs more than 7 times faster than the GoogLeNet during testing, which is critical and hence, highly desirable for real-time applications.

Across-channel vs within-channel filter sizes optimizatoin As mentioned in end of the methodology part, there are two settings for the filter size optimization: the across-channel (Eq. 3.24) and within-channel (Eq. 3.25) filter sizes optimization. As shown in Table 3.6, the comparison experiment is evaluated on the BP4D database in terms of F1 score as shown in Fig. 3.7. The experiment results show that the within-channel has better performance than the across-channel setting for most of the AUs, which agrees with our motivation that the different channels learn the different filter patterns and thus need the different filter sizes.

The filter sizes changing along with iteration: In order to having a better understanding the filter size optimization, Fig. 3.6 shows the filter sizes changing along with iterations on the BP4D database. All the filter sizes are initilized with 6×6. From the figure, the different AUs have the different trends along with the iterations. The trends are different because of the different initilization and different

Table 3.7 Performance comparison between Across-channel and within-channel filter sizes optimization on BP4D database in terms of F1 score.

AUs	%	Across-channel	Within-channel
AU1	23.1	0.345	**0.371**
AU2	17.9	0.303	**0.323**
AU4	22.7	0.415	**0.420**
AU6	46.0	0.729	**0.734**
AU7	52.6	0.649	0.649
AU10	59.6	**0.754**	0.745
AU12	55.8	0.805	**0.809**
AU14	52.1	0.562	**0.566**
AU15	18.0	**0.337**	0.336
AU17	32.6	0.563	**0.566**
AU23	17.0	0.398	**0.400**
AVE	-	0.533	0.538

AUs. There are steps changing for the red and blue color, when the filter sizes are out of range $[5, 7)$ and transformed to the next range. A step changing is implemented for the stabilization of the new expanded or shrinked filters.

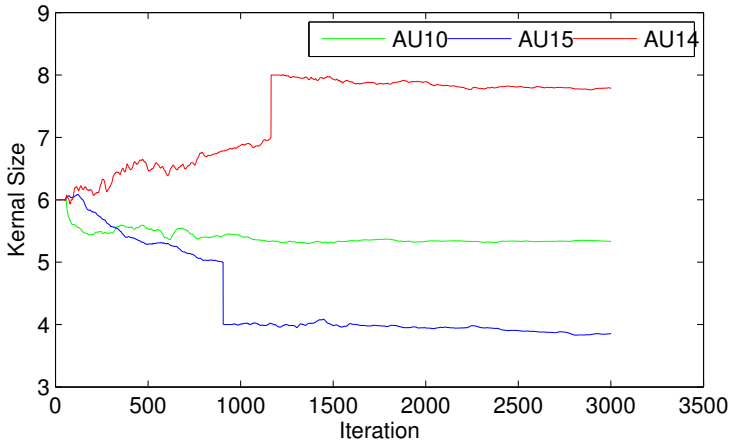

Figure 3.6 The kernel size changing along with the iteration

Comparison with state-of-the-art AU recognition methods: In addition to the baseline CNN, we further compare the proposed OFS-CNN with state-of-the-art methods, especially the CNN-based methods [17, 15, 80], on the four benchmark databases using the metrics that are common in those papers. As shown in Table 3.8,

Table 3.8 Performance comparison of state-of-the-art methods on four benchmark databases in terms of common metrics. ACC: Average classification rate.

CK (14AUs)		SEMAINE (6AUs)			BP4D (11AUs)			DISFA (10AUs)		
Methods	**ACC**	**Methods**	**F**	**2AFC**	**Methods**	**F1**	**2AFC**	**Methods**	**2AFC**	**ACC**
Gabor+SVM [5]	0.948	LGBP+SVM [66]	0.351	0.249	LGBP+SVM [66]	0.580	0.380	BGCS [59]	N/A	0.868
Gabor+DBN [65]	0.933	CNN [17]	0.341	N/A	CNN [17]	0.522	N/A	LPQ+SVM [24]	0.810	N/A
LBP+Geometry [18]	0.949	DLE-SIFT [72]	0.435	0.784	DLE-SIFT [72]	0.591	0.763	ML-CNN [15]	0.757	0.846
					DRML [80]	0.483	0.560	DRML [80]	0.523	N/A
CNN (baseline)	0.905	CNN (baseline)	0.363	0.737	CNN (baseline)	0.522	0.691	CNN (baseline)	0.843	0.800
OFS-CNN(AC)	0.920	OFS-CNN(AC)	0.382	0.746	OFS-CNN(AC)	0.533	0.704	OFS-CNN(AC)	0.842	0.825
OFS-CNN	0.923	**OFS-CNN**	0.386	0.762	**OFS-CNN**	0.538	0.711	**OFS-CNN**	0.848	0.830

the performance of the proposed OFS-CNN is better than that of the baseline CNN for all databases. OFS-CNN(AC) uses the across-channel filter optimization, and the OFS-CNN with within-channel has the better performance. A paired T-test is conducted and shows that the OFS-CNN statistically significantly outperforms the baseline CNN method for all metrics (p-values less than 0.05), except the 2AFC score on the DISFA dataset. Furthermore, it also beats the state-of-the-art CNN-based methods, i.e., the CNN [17] on the SEMAINE and BP4D databases [66], the DRML [80] on the BP4D [66] and the DISFA databases [43], and the ML-CNN [15] on the DISFA database [43] [1].

In addition, the OFS-CNN also achieves performance comparable to the other state-of-the-art methods using hand-crafted features. The proposed mehtod only uses the appearance information, while some hand crafted methods benefit from additional information such as landmarks[18] or explicitly modeling the relationship among AUs [65, 72]. Additionally, it is well known that deep learning is data hungry. AU coded images are limited and usually collected from a very small population. Although better performance has achieved by the hand-crafted features on most of the datasets, the deep learning methods do have a great potential to improve AU recognition when more AU-coded databases are available.

[1] Although the OFS-CNN has a slightly lower ACC than the ML-CNN [15], it has a much higher 2AFC score, which is more favorable to AU recognition with highly unbalanced positive/negative samples.

Comparison with state-of-the-art facial expression recognition methods: As mentioned before, the deep learning is data hungry and images on above AU-coded databases are collected from a small mount of subjects, all less than 30 subjects in the training stage. So the proposed method is evluated on the spontaneous facial expression database SFEW with thousands of subjects. There are 1,766 images including 958 images for training and 436 images for validation. They are all static images, so there are more than one thousands subjects. The CNN models of face expression recognition are fine-tuned from a CNN model pretrained on the Facial Expression Recognition (FER-2013) dataset [16] using SGD optimizer with a batch size of 300, momentum of 0.9, and a weight decay of 0.01. Dropout is applied to each FC layer with a prabability of 0.6.

Table 3.9 is the confusion matrix for the different expressions. The fear expression is easily confused with anger, becuase they both have the appearance of opening mouth and opening eyes. Furthermore, the proposed method within-channel OFS-CNN outperforms the baseline method with traditional fixed kernel size (46.62% vs 49.28%), and has the comparable performance with the state-of-the-art methods as shown in Table 3.10. The reported performance in our experiment is the average accuracy of 5 runs. FN2EN [10] emplyed 2.6 million face images to pre-train a face recognition VGG network and then use it to finetune the face expression network. IACNN employed the identity information together to improve the recognition performance to 50.98%. The comparison methods don't include the EmotiW 2015 winner [71] and runner-up [29] because they got the performance 56.40% with the fussion of many CNN models.

3.4 Chapter Summary

Traditional CNNs have a predefined and fixed integral filter sizes for each convolutional layer, which however, may be not optimal for all tasks as well as for all image

Table 3.9 Confusion matrix of the proposed OFS-CNN method evaluated on the SFEW [9] testing set. The ground truth and the predicted labels are given by the first column and the first row, respectively.

	An	Di	Fe	Ha	Ne	Sa	Su
An	**62.3%**	0%	2.6%	6.5%	9.1%	12%	7.8%
Di	4.4%	**0%**	4.4%	8.7%	26.1%	39.1%	17.4%
Fe	23.2%	0%	**0%**	10.7%	17.1%	30.0%	19.1%
Ha	5.5%	0%	0%	**84.9%**	1.4%	8.2%	0%
Ne	7.0%	0%	0%	3.5%	**57.0%**	30.2%	2.3%
Sa	11.1%	0%	1.3%	8.4%	19.0%	**46.5%**	13.6%
Su	19.3%	0%	5.3%	10.5%	28.1%	8.8%	**28.1%**

Table 3.10 Performance comparison on the validation set of SFEW database [9] in terms of the average accuracy of 7 expressions.

Method	Accuracy
AUDN [34]	26.14
STM-ExpLet [36]	31.73
Inception [46]	47.70
Mapped LBP [32]	41.92
BN [10]	39.55
VGG Fine-Tune [10]	41.23
Transfer Learning [48]	48.50
IACNN [45]	50.98
CNN (baseline)	46.62
OFS-CNN(AC)	49.11
OFS-CNN	**49.28**

resolutions. In contrast, the filter sizes are defined as continuous variables and can be learned from training data for all convolutional layers simultaneously through a novel OFS-CNN. Specifically, a forward-backward propagation algorithm is developed for the OFS-CNN to iteratively optimize the filter size while learning the convolution filters. Upper-bound and lower-bound filters are defined to facilitate the convolution operations with continuous-size filters. In addition, transformation operations are developed to accommodate the size changes of the filters. Furthermore, it has been shown that the proposed OFS-CNN has similar computational complexity compared with the traditional CNNs in the forward process and thus, during testing. In

addition, the proposed OFS-CNN is robust to the different filter size initializaitons. Compared with the GoogLeNet, the proposed method has slightly better performance and with very shallow network.

Experimental results on four benchmark AU databases and one spontaneous expression database have shown that the OFS-CNN outperforms the baseline CNNs with fixed filter sizes as well as the state-of-the-art CNN-based methods and, more importantly, achieves better or at least comparable performance to the baseline CNNs with the converged filter size found by exhaustive search. Furthermore, the OFS-CNN has been shown to be effective for automatically adapting filter sizes to different image resolutions. Experimental results on expression database SFEW with thousands of subjects have shown that the OFS-CNN also outperforms the baseline CNNs and have comparable performance to the state-of-the-art CNN-based method.

Chapter 4

Conclusion

In summary, two novel CNN works were proposed to handle the facial AU recognition challenges and improve the recognition rate, by substituting the traditional decision layer and convolutional layer with the incremental boosting layer and adaptive convolutional layer respectively.

First, a novel IB-CNN is proposed to handle the limited AU-coded training data and improve the AU recognition performance. The IB-CNN integrates boosting into the CNN via an incremental boosting layer that selects discriminative neurons and makes AU prediction. In addition, a novel loss function was proposed to fine-tune the IB-CNN. Experimental results on four benchmark AU databases have demonstrated that the IB-CNN outperforms the traditional CNN and has comparable performance to the state-of-the-art CNN-based methods for AU recognition. The improvement is more impressive for the AUs that have the lowest frequencies in the databases. The proposed IB-CNN has better and stable performance with different setting of learning rate and the number of input features, compared with traditional CNN.

Second, as far as we know all the current CNNs use predefined and fixed convolutional filter size. However, AUs activated by different facial muscles cause facial appearance changes at different scales and thus favor different filter sizes. A traditional strategy is to experimentally select the best kernel size in each convolutional layer for each AU, but it suffers from expensive training cost. We proposed a method to optimize the convolutional filter sizes by approximating the derivative of CNN loss with respect to the filter size. The experiments on four spontaneous AU-coded databases and one spontaneous expression database show the proposed method achieves better or at least comparable performance to the baseline convolutional layer with the best filter sizes found by exhaustive search. Furthermore, the OFS-CNN can learn and adjust the filter sizes from the different resolution of training images. Compared with the GoogLeNet, the proposed structure with OFS-CNN has slightly better performance and very shallow layers. In addition, the proposed method is robust to the

different filter sizes initialization. Since all the AU-coded databases only have a small mount of training subjects, usually less than 30, we also evaluation the proposed method with a shallow network on facial expression database SFEW. The experiment results show that proposed method apparently outperforms the baseline CNN and has comparable performance with the state-of-the-art methods.

Bibliography

[1] T. Almaev, A. Yüce, A. Ghitulescu, and M. Valstar, *Distribution-based iterative pairwise classification of emotions in the wild using LGBP-TOP*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), 2013, pp. 535–542.

[2] T. R. Almaev and M. F. Valstar, *Local Gabor binary patterns from three orthogonal planes for automatic facial expression recognition*, Proc. Int. Conf. on Affective Computing and Intelligent Interaction (ACII), Sept 2013, pp. 356–361.

[3] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, *Robust discriminative response map fitting with constrained local models*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3444–3451.

[4] T. Baltrusaitis, M. Mahmoud, and P. Robinson, *Cross-dataset learning and person-specific normalisation for automatic action unit detection*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), vol. 6, 2015, pp. 1–6.

[5] M. S. Bartlett, G. Littlewort, M. G. Frank, C. Lainscsek, I. Fasel, and J. R. Movellan, *Recognizing facial expression: Machine learning and application to spontaneous behavior*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 568–573.

[6] M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan, *Automatic recognition of facial actions in spontaneous expressions*, J. Multimedia **1** (2006), no. 6, 22–35.

[7] I. Bociu and I. Pitas, *A new sparse image representation algorithm applied to facial expression recognition*, Proc. Int. Workshop on Machine Learning for Signal Processing (MLSC), 2004, pp. 539–548.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, *Return of the devil in the details: Delving deep into convolutional nets*, Proc. British Machine Vision Conf. (BMVC), 2014.

[9] Abhinav Dhall, OV Ramana Murthy, Roland Goecke, Jyoti Joshi, and Tom Gedeon, *Video and image based emotion recognition challenges in the wild: Emotiw 2015*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), ACM, 2015, pp. 423–426.

[10] H. Ding, S. Zhou, and R. Chellappa, *Facenet2expnet: Regularizing a deep face recognition net for expression recognition*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), IEEE, 2017, pp. 118–126.

[11] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial action coding system: the manual*, Research Nexus, Div., Network Information Research Corp., Salt Lake City, UT, 2002.

[12] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, *Visualizing higher-layer features of a deep network*, Dept. IRO, Université de Montréal, Tech. Rep (2009).

[13] B. Fasel, *Head-pose invariant facial expression recognition using convolutional neural networks*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), 2002, pp. 529–534.

[14] B. Fasel, *Robust face analysis using convolutional neural networks*, Pattern Recognition **2** (2002), 40–43.

[15] S. Ghosh, E. Laksana, S. Scherer, and L. Morency, *A multi-label convolutional neural network approach to cross-domain action unit detection*, Proc. Int. Conf. on Affective Computing and Intelligent Interaction (ACII) (2015).

[16] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D. Lee, et al., *Challenges in representation learning: A report on three machine learning contests*, Proc. Int. Conf. on Machine learning (ICML), Springer, 2013, pp. 117–124.

[17] A. Gudi, H. E. Tasli, T. M. den Uyl, and A. Maroulis, *Deep learning based FACS action unit occurrence and intensity estimation*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2015.

[18] S. Han, Z. Meng, P. Liu, and Y. Tong, *Facial grid transformation: A novel face registration approach for improving facial action unit recognition*, Proc. Int. Conf. on Image Processing (ICIP), 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[20] G. E Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv preprint (2012).

[21] P. O. Hoyer, *Non-negative matrix factorization with sparseness constraints*, J. Machine Learning Research **5** (2004), 1457–1469.

[22] S. Jaiswal and M. F. Valstar, *Deep learning the dynamic appearance and shape of facial action units*, IEEE Workshop on Applications of Computer Vision (WACV), 2016.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, *Caffe: Convolutional architecture for fast feature embedding*, Proc. ACM Int. Conf. Multimedia, ACM, 2014, pp. 675–678.

[24] B. Jiang, B. Martinez, M. F. Valstar, and M. Pantic, *Decision level fusion of domain specific regions for facial action recognition*, Proc. Int. Conf. on Pattern Recognition (ICPR), 2014, pp. 1776–1781.

[25] B. Jiang, M.F. Valstar, and M. Pantic, *Action unit detection using sparse appearance descriptors in space-time video volumes*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2011.

[26] H. Jung, S. Lee, S. Park, I. Lee, C. Ahn, and J. Kim, *Deep temporal appearance-geometry network for facial expression recognition*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2015).

[27] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim, *Joint fine-tuning in deep neural networks for facial expression recognition*, Proc. Int. Conf. on Computer Vision (ICCV), 2015, pp. 2983–2991.

[28] T. Kanade, J. F. Cohn, and Y. Tian, *Comprehensive database for facial expression analysis*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2000, pp. 46–53.

[29] B-K Kim, H. Lee, J. Roh, and S-Y Lee, *Hierarchical committee of deep cnns with exponentially-weighted decision fusion for static facial expression recognition*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), 2015, pp. 427–434.

[30] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature **401** (1999), no. 6755, 788–791.

[31] G. Levi and T. Hassner, *Age and gender classification using convolutional neural networks*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 34–42.

[32] Gil Levi and Tal Hassner, *Emotion recognition in the wild via convolutional neural networks and mapped binary patterns*, Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, 2015, pp. 503–510.

[33] Y. Lin, M. Song, D.T.P. Quynh, Y. He, and C. Chen, *Sparse coding for flexible, robust 3d facial-expression synthesis*, Computer Graphics and Applications **32** (2012), no. 2, 76–88.

[34] M. Liu, S. Li, S. Shan, and X. Chen, *AU-aware deep networks for facial expression recognition*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2013, pp. 1–6.

[35] M. Liu, S. Li, S. Shan, R. Wang, and X. Chen, *Deeply learning deformable facial action parts model for dynamic expression analysis*, Proc. Asian Conf. on Computer Vision (ACCV), 2014.

[36] M. Liu, S. Shan, R. Wang, and X. Chen, *Learning expressionlets on spatio-temporal manifold for dynamic facial expression recognition*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1749–1756.

[37] P. Liu, S. Han, Z. Meng, and Y. Tong, *Facial expression recognition via a boosted deep belief network*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1805–1812.

[38] P. Liu, S. Han, and Y. Tong, *Improving facial expression analysis using histograms of log-transformed nonnegative sparse representation with a spatial pyramid structure*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2013, pp. 1–7.

[39] W. Liu, C. Song, and Y. Wang, *Facial expression recognition based on discriminative dictionary learning*, Proc. Int. Conf. on Pattern Recognition (ICPR), 2012.

[40] S. Lucey, A. B. Ashraf, and J. Cohn, *Investigating spontaneous facial action recognition through AAM representations of the face*, Face Recognition Book (K. Kurihara, ed.), Pro Literatur Verlag, Mammendorf, Germany, April 2007.

[41] M. H. Mahoor, M. Zhou, K. L. Veon, S. M. Mavadati, and J. F. Cohn, *Facial action unit recognition with sparse representation*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2011, pp. 336–342.

[42] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, *Subject independent facial expression recognition with robust face detection using a convolutional neural network*, IEEE Trans. on Neural Networks **16** (2003), no. 5, 555–559.

[43] S Mohammad Mavadati, Mohammad H Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F Cohn, *Disfa: A spontaneous facial action intensity database*, IEEE Trans. on Affective Computing **4** (2013), no. 2, 151–160.

[44] D. Medera and S. Babinec, *Incremental learning of convolutional neural networks.*, IJCCI, 2009, pp. 547–550.

[45] Z. Meng, P. Liu, J. Cai, S. Han, and Y. Tong, *Identity-aware convolutional neural network for facial expression recognition*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), IEEE, 2017, pp. 558–565.

[46] A. Mollahosseini, D. Chan, and M. H. Mahoor, *Going deeper in facial expression recognition using deep neural networks*, IEEE Workshop on Applications of Computer Vision (WACV), 2015.

[47] J. Nagi, G. A Di Caro, A. Giusti, F. Nagi, and L. M Gambardella, *Convolutional neural support vector machines: hybrid visual pattern classifiers for multi-robot systems*, Proc. Int. Conf. on Machine Learning and Applications (ICMLA), 2012, pp. 27–32.

[48] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler, *Deep learning for emotion recognition on small datasets using transfer learning*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), 2015, pp. 443–449.

[49] B. A. Olshausen and D. J. Field, *Emergence of simple-cell receptive field properties by learning a sparse code for natural images*, Nature **381** (1996), no. 6583, 607–609.

[50] M. Pantic, A. Pentland, A. Nijholt, and T. S. Huang, *Human computing and machine understanding of human behavior: A survey*, Artificial Intelligence for

Human Computing (T. S. Huang, A. Nijholt, M. Pantic, and A. Pentland, eds.), Lecture Notes in Artificial Intelligence, Springer Verlag, London, 2007.

[51] M Ranzato, J. Susskind, V. Mnih, and G. Hinton, *On deep generative models with applications to recognition*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2857–2864.

[52] S. Reed, K. Sohn, Y. Zhang, and H. Lee, *Learning to disentangle factors of variation with manifold interaction*, Proc. Int. Conf. on Machine learning (ICML), 2014.

[53] S. Rifai, Y. Bengio, A. Courville, P. Vincent, and M. Mirza, *Disentangling factors of variation for facial expression recognition*, Proc. European Conf. on Computer Vision (ECCV), 2012, pp. 808–822.

[54] E. Sariyanidi, H. Gunes, and A. Cavallaro, *Automatic analysis of facial affect: A survey of registration, representation and recognition*, IEEE Trans. on Pattern Analysis and Machine Intelligence **37** (2015), no. 6, 1113–1133.

[55] M. A. Sayette, J. F. Cohn, J. M. Wertz, M. A. Perrott, and D. J. Parrott, *A psychometric evaluation of the facial action coding system for assessing spontaneous expression*, J. Nonverbal Behavior **25** (2001), no. 3, 167–185.

[56] K. Scherer and P. Ekman, *Handbook of methods in nonverbal behavior research*, Cambridge University Press, Cambridge, UK, 1982.

[57] T. Sénéchal, V. Rapp, H. Salam, R. Seguier, K. Bailly, and L. Prevost, *Combining AAM coefficients with LGBP histograms in the multi-kernel SVM framework to detect facial action units*, Proc. Int. Conf. on Automatic Face and Gesture Recognition Workshop (FGW), 2011, pp. 860 – 865.

[58] C. Shan, S. Gong, and P.W. McOwan, *Facial expression recognition based on Local Binary Patterns: A comprehensive study*, J. Image and Vision Computing **27** (2009), no. 6, 803–816.

[59] Y. Song, D. McDuff, D. Vasisht, and A. Kapoor, *Exploiting sparsity and co-occurrence structure for action unit recognition*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2015.

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabi-

novich, *Going deeper with convolutions*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[61] Y. Tang, *Deep learning using linear support vector machines*, Proc. Int. Conf. on Machine learning (ICML), 2013.

[62] A Teixeira Lopes, E de Aguiar, and T Oliveira-Santos, *A facial expression recognition system using convolutional networks*, Graphics, Patterns and Images (SIBGRAPI), 2015 28th SIBGRAPI Conference on, IEEE, 2015, pp. 273–280.

[63] Y. Tian, T. Kanade, and J. F. Cohn, *Evaluation of Gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), May 2002, pp. 229–234.

[64] Y. Tong, J. Chen, and Q. Ji, *A unified probabilistic framework for spontaneous facial action modeling and understanding*, IEEE Trans. on Pattern Analysis and Machine Intelligence **32** (2010), no. 2, 258–274.

[65] Y. Tong, W. Liao, and Q. Ji, *Facial action unit recognition by exploiting their dynamic and semantic relationships*, IEEE Trans. on Pattern Analysis and Machine Intelligence **29** (2007), no. 10, 1683–1699.

[66] M. Valstar, J. Girard, T. Almaev, G. McKeown, M. Mehu, L. Yin, M. Pantic, and J. Cohn, *FERA 2015 - second facial expression recognition and analysis challenge*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG) (2015).

[67] M. F. Valstar, M. Mehu, B. Jiang, M. Pantic, and K. Scherer, *Meta-analyis of the first facial expression recognition challenge*, IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics **42** (2012), no. 4, 966–979.

[68] J. Whitehill, M. S. Bartlett, G. Littlewort, I. Fasel, and J. R. Movellan, *Towards practical smile detection*, IEEE Trans. on Pattern Analysis and Machine Intelligence **31** (2009), no. 11, 2106–2111.

[69] P. Yang, Q. Liu, and D. N. Metaxas, *Boosting coded dynamic features for facial action units and facial expression recognition*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2007, pp. 1–6.

[70] Z. Ying, Z. Wang, and M. Huang, *Facial expression recognition based on fusion of sparse representation*, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence (D.-S. Huang, X. Zhang, C.A.

Reyes García, and L. Zhang, eds.), Lecture Notes in Computer Science, 2010, pp. 457–464.

[71] Z. Yu and C. Zhang, *Image based static facial expression recognition with multiple deep network learning*, Proc. Int. Conf. on Multimodal Interfaces (ICMI), 2015, pp. 435–442.

[72] A. Yuce, H. Gao, and J. Thiran, *Discriminant multi-label manifold embedding for facial action unit detection*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 2015.

[73] S. Zafeiriou and M. Petrou, *Nonlinear non-negative component analysis algorithms*, IEEE Trans. on Image Processing **19** (2010), no. 4, 1050–1066.

[74] S. Zafeiriou and M. Petrou, *Sparse representations for facial expressions recognition via L1 optimization*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 32–39.

[75] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, Proc. European Conf. on Computer Vision (ECCV), 2014, pp. 818–833.

[76] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, *A survey of affect recognition methods: Audio, visual, and spontaneous expressions*, IEEE Trans. on Pattern Analysis and Machine Intelligence **31** (2009), no. 1, 39–58.

[77] Y. Zhang and Q. Ji, *Active and dynamic information fusion for facial expression understanding from image sequences*, IEEE Trans. on Pattern Analysis and Machine Intelligence **27** (2005), no. 5, 699–714.

[78] Z. Zhang, M. Lyons, M. Schuster, and S. Akamatsu, *Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron*, Proc. Int. Conf. on Automatic Face and Gesture Recognition (FG), 1998, pp. 454–459.

[79] G. Zhao and M. Pietiäinen, *Dynamic texture recognition using local binary patterns with an application to facial expressions*, IEEE Trans. on Pattern Analysis and Machine Intelligence **29** (2007), no. 6, 915–928.

[80] K. Zhao, W. Chu, and H. Zhang, *Deep region and multi-label learning for facial action unit detection*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3391–3399.

[81] R. Zhi, M. Flierl, Q. Ruan, and W.B. Kleijn, *Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition*, IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics (2010), no. 99, 1–15.

[82] L. Zhong, Q. Liu, P. Yang, J. Huang, and D.N. Metaxas, *Learning multiscale active facial patches for expression analysis*, IEEE Trans. on Cybernetics **45** (2015), no. 8, 1499–1510.